TURING 图灵程序设计丛书

[PACKT] PUBLISHING



Design Mobile-First Responsive Websites with Bootstrap 3

Bootstrap实战

Bootstrap Site Blueprints

[美] David Cochran 著　　李松峰 译
Ian Whitley

中国工信出版集团　　人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

David Cochran（作者） 现为俄克拉荷马州卫斯理大学副教授，自2005年以来一直讲授交互设计。他重视最佳实践，不喜欢走捷径，推崇Web标准。2012年，David在webdesign.tutsplus.com上开设了Bootstrap 2.0系列教程专栏。他还在Packt Publishing出版过一本小书*Twitter Bootstrap Web Development How-To*。闲暇时间，David会在自己的网站alittlecode.com上写写博客。他还是媒体、设计和咨询公司BitBrilliant的负责人。

Ian Whitley（作者）自幼爱好写作。2010年，他对Web开发产生了浓厚兴趣，并投身其中。他很早就开始使用Twitter Bootstrap，而且在David Cochran帮助下很快就掌握了这个框架，并将其应用到了很多项目中。目前，他多数情况下会使用Bootstrap为客户创建WordPress模板。Ian是BitBrilliant公司开发主管。

李松峰（译者） 2006年起投身翻译，译著30余部，包括《JavaScript高级程序设计》《简约至上》等畅销书。2008年进入出版业，目前在图灵公司从事技术图书策划和审稿工作，并担任《Web+DB Press中文版》杂志编委。他经常参加社区活动，曾在W3ctech 2012 Mobile上分享“Dive into Responsive Web Design”。2013年1月应邀在金山网络分享“响应式Web设计”，3月应邀在奇虎360分享“JS的国”。在2014 DevFest Beijing分享了“理解响应式Web设计”。

图灵程序设计丛书



Designing Mobile-First Responsive Websites with Bootstrap 3

Bootstrap实战

Bootstrap Site Blueprints

[美] David Cochran 著　　李松峰 译
Ian Whitley

人民邮电出版社
北京

图书在版编目（C I P）数据

Bootstrap实战 / (美) 科克伦 (Cochran,D.) ,
(美) 惠特利 (Whitley,I.) 著 ; 李松峰译. -- 北京 :
人民邮电出版社, 2015.5
（图灵程序设计丛书）
ISBN 978-7-115-38887-2

Ⅰ. ①B… Ⅱ. ①科… ②惠… ③李… Ⅲ. ①网页制
作工具 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2015)第072099号

内 容 提 要

Bootstrap 是前端开发中应用非常广泛的一个框架，最早是 Twitter 公司内部的一个工具，开源之后迅速得到了各方的认可。本书基于最新的 Bootstrap 3 撰写，在简单介绍了安装与配置之后就直奔主题，分别讨论了个人作品站点、WordPress 主题、企业网站、电子商务网站和单页营销网站等几个最具代表性的应用案例，结合这些案例细致地剖析了 Bootstrap 还有 LESS 的使用方式和技巧。

本书适合所有前端开发人员及个人网站设计者阅读参考。

◆ 著　　[美] David Cochran [美] Ian Whitley
译　　李松峰
责任编辑　岳新欣
责任印制　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　　　印刷

◆ 开本：800×1000　1/16
印张：15
字数：273千字　　　　2015年 5 月第 1 版
印数：1－3 500册　　　　2015年 5 月北京第 1 次印刷

著作权合同登记号　图字：01-2014-8582号

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

版 权 声 明

前　　言

2011 年 8 月，Twitter Bootstrap 横空出世。如今，这个被称为 Bootstrap 的框架，俨然已成为前端设计领域最受欢迎的辅助技术。Bootstrap 3 又是一个里程碑式的版本，增加了移动优先的响应式网格、新的强大的 LESS 混入，而且还针对现代浏览器对核心代码进行了优化。

本书详细介绍 Bootstrap 的使用方法。全书简明易懂，循序渐进，让读者时时处处体验到自定义和重编译 Bootstrap 的 LESS 文件的强大威力，同时掌握应用 Bootstrap 的 JavaScript 插件设计专业用户界面的技巧。

这本书并不局限于 Bootstrap。Bootstrap 只是一个工具，一个达到目标的手段。学习完这本书之后，读者将成为一位更加高效、熟练的 Web 设计师。

本书内容

第 1 章“初识 Bootstrap”，教给大家如何下载 Bootstrap，如何基于 HTML5 Boilerplate 设置站点模板，并且掌握把 Bootstrap 的 LESS 文件编译为 CSS 的方法。

第 2 章“作品展示站点”，开始学习创建简单的个人作品展示网站，包括全宽的传送带切换效果，三栏文本布局，以及使用 Font Awesome 的字体图标——通过自定义 Bootstrap 的 LESS 文件。

第 3 章“WordPress 主题”，学习把第 2 章的个人作品展示站点转换成一个 WordPress 主题。这一章要利用有名的 Roots Theme，根据我们的需要，还会涉及自定义模板文件、LESS、CSS 和 JavaScript。

第 4 章“企业网站”，将通过创建一个企业级网站学习如何创建复杂的页头区，添加下拉菜单和实用导航，以及构建复杂的三栏布局和四栏页脚，同时还要确保所有这些内容具有完全的响应能力。

第 5 章“电子商务网站”，带领大家探索商品展示页面的设计，学会在复杂的响应式网格中控制多行商品。与此同时，还要实现一个响应式的商品筛选工具。

第 6 章“单页营销网站”，这一章教给大家构建时尚的单页营销网站，包括带高清图的大

字欢迎语、带大图标的商品功能列表、图片墙式的用户评论区，以及三个精美的价目表，最终完成一个支持动态滚动导航的漂亮的单页网站。

附录 A“优化站点资源”，以第 2 章的网站资源为例，介绍了为最终部署网站而优化 Bootstrap LESS/CSS 及 JavaScript 等静态资源的基本流程和工具。这个附录对所有 Bootstrap 项目都一样有用。

附录 B“实现响应式图片”，介绍最前沿的响应式图片解决方案 Picturefill，仍然以第 2 章项目中的作品图片传送带为例。这个附录对其他项目也一样有用。

附录 C“让传送带支持手势”，介绍在 Bootstrap 的传送带中使用一款先进的插件 Hammer.js，实现传送带图片的轻扫切换功能。

本书要求

要完成本书各章的项目，需要安装下列软件：

- 现代浏览器（包括 Internet Explorer 8 及以上版本）；
- 代码编辑器；
- LESS 编译器，其 less.js 的版本至少是 1.3.3。

读者对象

本书适合已经熟练掌握 HTML 和 CSS 基础的读者，最好是熟悉规范的 HTML5 和样式表的写法。了解 JavaScript 的基本知识，包括 Bootstrap 的 jQuery 插件用法更好。本书经常会涉及自定义 LESS 文件，因此熟悉 LESS 的读者会感觉更轻松一些。不过，即便从来没有使用过 LESS，本书由浅入深的介绍，也会让你顺利入门。

本书约定

本书正文中会涉及很多种版式，以区分不同的信息。以下是这些版式的解释。

正文中的代码将以代码体印刷，比如：

```
<FilesMatch "\.(ttf|otf|eot|woff)$">
  <IfModule mod_headers.c>
    Header set Access-Control-Allow-Origin "*"
  </IfModule>
</FilesMatch>
```

新术语和重要的词汇将以楷体印刷。截屏、对话框或菜单中的词汇，会保留英文原文。比如，“请单击 Download 按钮”。



这个图标表示警告或需要特别注意的内容。



这个图标表示提示或技巧。

读者反馈

欢迎提出反馈，你对本书有任何想法，喜欢它什么，不喜欢它什么，请让我们知道。要写出真正对大家有帮助的图书，读者的反馈很重要。

一般的反馈，请发送电子邮件至 feedback@packtpub.com，并在邮件主题中包含书名。

如果你有某个主题的专业知识，并且有兴趣写成或帮助促成一本书，请参考我们的作者指南 http://www.packtpub.com/authors。

客户支持

现在，你是一位令我们自豪的 Packt 图书的拥有者，我们会尽全力帮你充分利用你手中的书。

下载示例代码[①]

你可以用你的账户从 http://www.packtpub.com 下载所有已购买 Packt 图书的示例代码文件。如果你从其他地方购买本书，可以访问 http://www.packtpub.com/support 并注册，我们将通过电子邮件把文件发送给你。

勘误表[②]

虽然我们已尽力确保本书内容正确，但出错仍旧在所难免。如果你在我们的书中发现错误，不管是文本还是代码，希望能告知我们，我们不胜感激。这样做，你可以使其他读者免受挫败，帮助我们改进本书的后续版本。如果你发现任何错误，请访问 http://www.packtpub.com/submit-errata 提交，选择你的书，点击勘误表提交表单的链接，并输入详细说明。勘误一经核实，你的提交将被接受，此勘误将上传到本公司网站或添加到现有勘误表。从 http://www.packtpub.com/support 选择书名就可以查看现有的勘误表。

① 读者可以直接访问本书中文版页面，下载本书项目的源代码：http://ituring.cn/book/1418。——编者注

② 对中文版的勘误，请读者直接到本书中文版页面提交：http://ituring.cn/book/1418。——编者注

侵权行为

版权材料在互联网上的盗版是所有媒体都要面对的问题。Packt 非常重视保护版权和许可证。如果你发现我们的作品在互联网上被非法复制，不管以什么形式，都请立即为我们提供位置地址或网站名称，以便我们可以寻求补救。

请把可疑盗版材料的链接发到 copyright@packtpub.com。

非常感谢你帮助我们保护作者，以及保护我们给你带来有价值内容的能力。

问题

如果你对本书内容存有疑问，不管是哪个方面，都可以通过 questions@packtpub.com 联系我们，我们将尽最大努力来解决。

目　　录

初识 Bootstrap

1

作为 Web 前端开发框架，Bootstrap 的流行很容易理解。它为大多数标准的 UI 设计场景提供了用户友好、跨浏览器的解决方案。它现成可用且经受了社区考验的 HTML 标记、CSS 样式及 JavaScript 行为的组合，极大提高了 Web 前端界面的开发效率，创造了令人愉悦效果。有了这些基本的元素，开发人员就有了构建自己定制方案的坚实基础。

不过，流行、高效、有效也不一定都是好事。由于工具便捷而导致人们养成坏习惯的例子屡见不鲜。然而，Bootstrap 却没有这个问题，至少不一定存在这个问题。从它面世就一直关注它的人都知道，它的早期版本和更新有时候会更侧重实际效率，而非最佳实践。事实上，一些最佳实践不管是语义标记还是移动优先的设计，抑或资源性能优化，都需要额外的时间和精力才能实现。

1.1 数量和质量

运用得当的情况下，我认为 Bootstrap 无论从质量还是效率角度说，都是 Web 开发社区的一大福利。随着越来越多的开发者使用这个框架，越来越多的人也加入了这个社区，从而逐步接受了前沿的最佳实践。Bootstrap 从一开始就鼓励可靠、成熟的及面向未来的 CSS 方案，比如 Nicholas Galagher 的 Normalize.css 和用 CSS3 方案解决图片过多的问题。此外，它也支持 HTML5 语义标记。

1.1.1 与时俱进

Bootstrap v2.0 发布之后，响应式设计随之成为主流，其界面元素也做到了跨设备响应（包括桌面、平板和手机）。

现在，Bootstrap v3.0 也发布了，其功能愈加完善，包括：

- 移动优先的响应式网格；
- 基于 Web 字体的图标，适用于移动及高密度屏幕；
- 不再支持 IE7，标记和样式更加简洁高效。

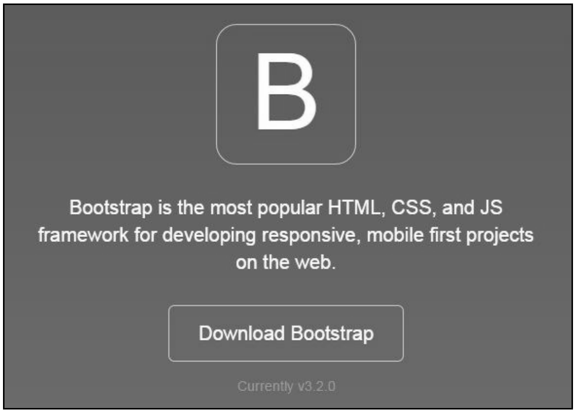
1.1.2 LESS 的威力

简洁的 CSS（LESS）值得重点说一下。从单纯给标记添加类深入到自定义 Bootstrap 的 LESS 文件，大幅提高了我们的工作效率和控制力。在 Bootstrap 默认的样式表基础之上，开发人员可以发挥自己的创造力，定制出自己的核心内容。

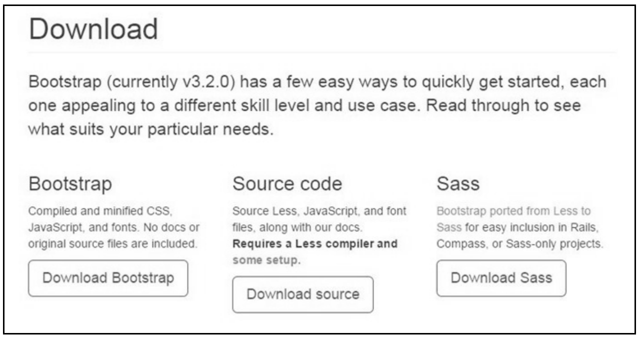
换句话说，Bootstrap 确实非常强大。我会在本书中展示其令人兴奋的特性、效率和最佳实践，告诉大家如何利用它做出漂亮、用户友好的界面。

1.2 下载 Bootstrap

下载 Bootstrap 的途径很多，但通过这些途径下载的文件并不完全一样。为了后面考虑，我们必须保证下载到 LESS 文件，因为这些文件可以为我们提供定制和创意的基础。在这里，我们直接溯本求源，打开 GetBootstrap.com。



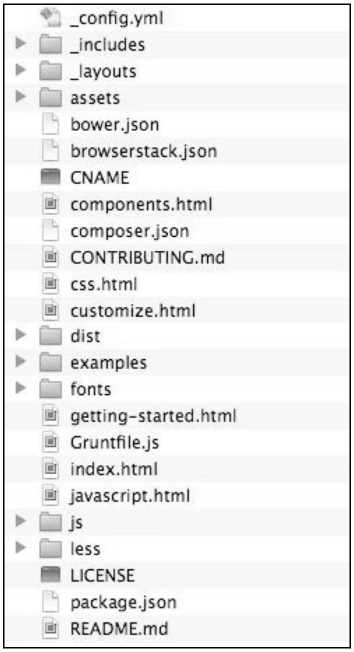
打开网站，一眼就能看到大大的“Download Bootstrap”按钮。翻译本书时，最新的版本为 v3.2.0。点击这个按钮，进入下载页面：



可以看到中间那个“Download source”按钮，点击下载即可。此外，也可以访问 GitHub 上的项目链接（https://github.com/twbs/bootstrap），点击“Download ZIP”按钮。为方便创建示例，本书采用 Bootstrap v3.0.2 作为示范，这个版本可以在 GitHub 上找到并下载：https://github.com/twbs/bootstrap/releases。

下载后的文件

下载了 Bootstrap 的源文件之后，应该能看到类似下图所示的文件结构：



没错，文件不少，但我们并不需要那么多。无论如何，这里包含了 Bootstrap 所能提供的一切。

需要说明的是，下载到的文件中的实际内容可能会随着时间推移而有所变化，但主要内容通常不会变。比如，在 less 文件夹中，可以找到所有重要的 LESS 文件，它们是本书所有项目的基础。另外，js 文件夹中包含的是 Bootstrap 的插件，可供我们选择使用。

假如你只需要 Bootstrap 默认提供的预编译的 CSS 或 JavaScript 文件（bootstrap.css 或 bootstrap.min.js），也可以在 dist 文件夹中找到它们。在这个版本的源文件里，还有一个 examples 文件夹，其中包含示例 HTML 模板。我们第一个项目的模板文件夹就会基于其中一个示例来创建。

1.3　准备项目模板文件夹

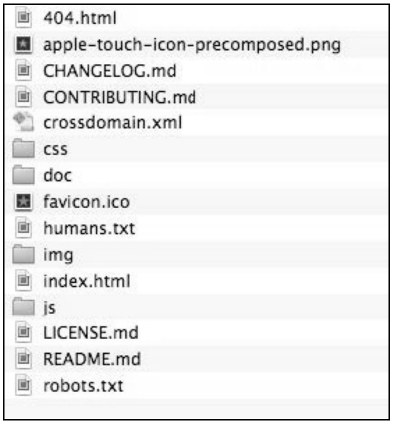
接下来，我们为第一个项目创建一个文件夹以及一些基本的文件。为此，我们还要用到 HTML5 样板文件 HTML5 Boilerplate（H5BP），然后把 Bootstrap 的有用文件复制过去。

1.3.1　下载 H5BP

在浏览器中打开 h5bp.com。这是一个短链接，服务器解析后会重定向到 H5BP 的主文档页面。可以在这个页面直接下载 H5BP，也可以单击 SOURCE CODE 链接，在 GitHub 上下载。

解压下载到的 ZIP 文件，并把文件夹重命名为 Project Template 1。

在这个文件夹中，可以看到类似下图所示的目录结构：



如果你的文件系统没有显示隐藏的文件，可能看不到.htaccess 文件。在我的电脑上，我是通过自己的 FTP 客户端打开这个本地文件夹，从而看到.htaccess 文件的。

1.3.2　删除不必要的样板文件

删除下列只与 H5BP 相关的文件夹和文件：

- 因为稍后我们要使用 LESS 创建自己的 CSS 文件，所以先删除 css 文件夹。
- CHANGELOG.md。
- CONTRIBUTING.md。
- doc 文件夹及其中内容。

1.3.3 理解样板中的.htaccess 文件

如果你还从未看过 H5BP 中的.htaccess 文件，建议先看看 H5BP 的文档（http://h5bp.com）。当然，这个文件本身也有详细注释，可以用编辑器打开从头到尾看一遍。这个文件中的内容不一定全都有用，这取决于你的主机设置和站点需求。不过这个文件的一个主要用途是保证站点性能最优。希望大家能够认真对待它，多听听高手的建议，然后根据需要配置它。就我而言，因为我的主机提供商会帮我处理这些事，所以就不需要这个.htaccess 文件了。

1.3.4 更新必要的样板文件

样板中的下列文件提供了项目的标准信息，根据需要你可以更新它们、直接使用它们，也可以放那儿不管，完全取决于你。

- humans.txt：这个文件记载贡献者，H5BP、Bootstrap 的，还有其他贡献者。
- LICENSE.md：在 H5BP 许可前面，加上你基于该许可构建的网站的许可信息，在 H5BP 许可之后，加上 Bootstrap 以及其他站点中用到的重要的库的许可信息。
- README.md：更新这个文件，加上自己项目的说明信息。

1.3.5 更新站点桌面和触摸设备图标

不要忘了用自己项目的图标替换 Boilerplate 默认的图标文件，包括以下图标。

- apple-touch-icon-precomposed.png：为确保所有移动设备（包括高像素密度屏幕设备）都有最佳效果，这个图标应该是 144px 见方的（样板文件中的是 152px 见方的）。
- favicon.ico：32px 见方的图标文件。

> 之前版本的样板文件曾经包含过多达 6 种尺寸的触摸设备图标。最近，这个做法有所改变。因为大图标可用于所有相关设备，而由此造成的性能损失非常非常小。为了减少开发人员制作图标的工作量，现在就只留下一个触摸设备图标了。如果你对相关的讨论感兴趣，可以参考这个链接：https://github.com/h5bp/html5-boilerplate/issues/1367#ref-pullrequest-18787780。

1.4 加入 Bootstrap 文件

现在，我们可以考虑把 Bootstrap 的文件弄到 Project Template 1 文件夹里来了。我们会从 Bootstrap 提供的一大堆文件里选出需要的部分。为了加快进度，强烈建议大家分别在两个窗口中打开 Bootstrap 3 文件夹和项目的模板文件夹，以便于比较和拖放。

1.4.1　字体

从 Bootstrap 的主文件夹中，把 fonts 文件夹复制粘贴到 Project Template 1 文件夹中。这个文件夹里包含着 Bootstrap 附带的重要的 Glyphicon 字体。（如果你之前没用过图标字体，相信你会喜欢的。）

保险起见，建议大家再在这个文件夹里放一个跨域友好的.htaccess 文件。为什么？因为随着越来越多的 CDN（Content Delivery Network，内容分发网络）为你的网站缓存静态资源，你会发现如果没有这个文件，某些浏览器会拒绝识别你的 Web 字体。（注意，H5BP 的.htaccess 文件中包含了解决这个问题的代码。而我们需要做的，就是保证即使站点根目录下没有放 H5BP 的.htaccess 文件，也不会出现字体问题。）

在代码编辑中创建一个新文件，添加以下代码：

```
<FilesMatch "\.(ttf|otf|eot|woff)$">
  <IfModule mod_headers.c>
    Header set Access-Control-Allow-Origin "*"
  </IfModule>
</FilesMatch>
```

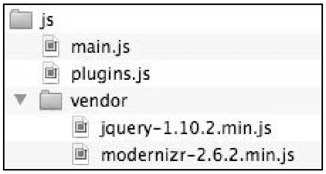
> **下载示例代码**
>
> Packt 所有图书的示例代码都可以登录 http://www.packtpub.com 下载。如果没有注册过，需要注册一个账号，下载地址会发到你的注册邮箱里。

把这个新文件直接保存到 fonts 文件夹中，并将其命名为.htaccess。（注意，如果是在本地计算机中，你的操作系统可能不会显示这个文件。假如你不知道怎么让操作系统显示隐藏的文件，可以借助 FTP 客户端，设置客户端的首选项能查看隐藏的文件，然后使用它的浏览器窗口来查看这个本地文件夹。）

保存好之后，这个.htaccess 就在你的 fonts 文件夹中了，它能够确保无论你的站点使用什么 CDN 服务，所有浏览器都可以使用你的 Web 字体。

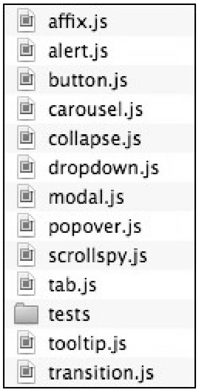
1.4.2　JavaScript

好，接下来要加入 Bootstrap 的 JavaScript 文件。H5BP 的文件夹中已经包含了一个放置 JavaScript 文件的文件夹（名为 js），在这个文件夹里，可以看到 4 个文件——有 2 个位于名为 vendor 的子文件夹内，如下面的截图所示：



Bootstrap 的插件都基于 jQuery，而 H5BP 已经为我们准备好了。除了 jQuery，我们还看到了 Modernizr 脚本。简单介绍一下，Modernizr 包含的是 HTML5“垫片”（shiv）脚本，可以让 IE8 支持 HTML5 的分区（section）元素。因为我们这个项目打算支持 IE8，所以也用得着它。除此之外，Modernizr 还可以让我们更方便地检测特定浏览器的能力，比如 CSS 3D 变换（要了解更多信息，请参考其文档，地址为：http://modernizr.com/docs/）。下一章，我们会用到 Modernizr 的特性检测功能。

接下来我们实际要做的，就是把 Bootstrap 的插件脚本都弄进来。首先，我们把它们以单个文件的形式复制过来。在 Project Template 1 文件夹的子文件夹 js 中，再创建一个名为 bootstrap 的文件夹，然后把 Bootstrap 的 js 文件夹中的脚本文件都复制过来。下面截图显示了 Bootstrap 随带的插件，每个插件一个文件：



把这些插件文件集中保存到新建的 js/bootstrap 文件夹，便于优化网站性能，即可以按需选用插件、排除其他文件并缩减文件大小。

在开发期间，保持所有 Bootstrap 的插件都可用也是一个办法。这样，如果你想添加个折叠、提示或者传送带效果，都可以信手拈来。我们在此选择自己的做法。

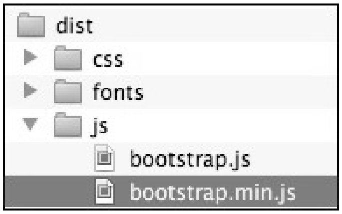
H5BP 采用的方法是把所有插件代码都复制到一个 plugins.js 模板文件中。这是结束开发之后的最佳做法，因为这样可以减少 HTTP 请求，加快站点加载速度。（换句话说，加载一个 80 KB 的文件，比加载 4 个 20 KB 的文件速度更快。）

我们在这个项目的开发过程中，也将采用相同的方法。我们要做的只是把必要的插件代

码复制到 plugins.js0 文件中。所以下面我们就要找到 Bootstrap 中包含插件代码的大文件。

> 或许有读者喜欢在开发期间分别在标记中链接用到的单个插件，最终再把它们合并成一个。如果你愿意，大可这样做，忽略我们下面的内容即可。

打开 Bootstrap 文件夹中包含分发文件的 dist 文件夹。在这个文件夹中的 js 文件夹里，包含着 bootstrap.js 和 bootstrap.min.js，它们就是包含 Bootstrap 所有插件代码的大文件。考虑到这一章的练习不用编辑插件代码，所以我们可以直接使用压缩后的版本。



找到图中选中的文件后，完成下列步骤：

(1) 在编辑器中打开 bootstrap.min.js；
(2) 全选代码，包括开关的注释，然后复制；
(3) 打开新项目文件夹中的 plugins.js；
(4) 把 Bootstrap 的插件代码粘贴到`// Place any jQuery/helper plugins in here`注释下面。结果看起来如下所示（这里当然省略了很多后续代码）。

```
// Place any jQuery/helper plugins in here.

/**
* bootstrap.js v3.0.0 by @fat and @mdo
* Copyright 2013 Twitter Inc.
* http://www.apache.org/licenses/LICENSE-2.0
*/
if(!jQuery)throw new Error("Bootstrap requires jQuery");+function(a){"use strict";
...
```

(5) 保存并退出。

这样就把 Bootstrap 的插件都准备好了！

> 保留 Bootstrap 插件及其他插件开头的注释，就保留了插件的来源信息，这些信息也是许可的基本内容。此外，我们因此也更容易搜索到相关插件。比如，在发布之前优化代码时，就需要把现在的压缩版本替换成只包含用到的插件的压缩版本。保留注释到时候就会派上用场。

1.4.3 暂时不考虑 CSS 文件

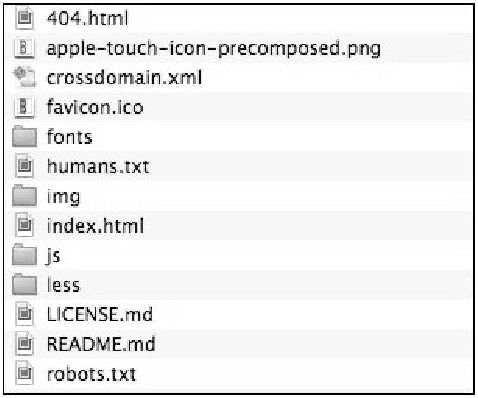
在后面的项目中，我们会使用 LESS 创建自定义的 Bootstrap CSS 文件。下一章开始就会这么做，现在先不用考虑 CSS。

1.4.4 复制 LESS 文件

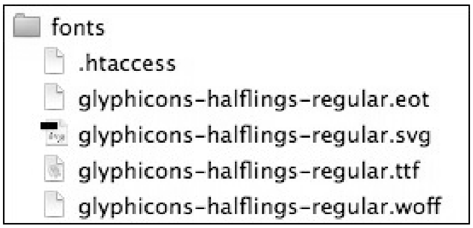
下面我们把 Bootstrap 中重要的 LESS 文件复制过来：把 bootstrap/less 文件夹复制到 Project Template 1 文件夹中。

1.5 大盘点

现在，Project Template 1 文件夹应该如下图所示：



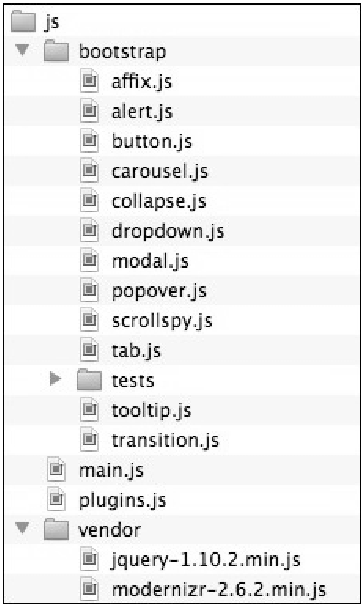
而 fonts 文件夹，包括新创建的.htaccess 文件在内，应该如下图所示：



> 我一直在用自己的 FTP 客户端来查看文件，并将其设置为显示隐藏的文件。如果你没有像我这样，有可能看不到隐藏的.htaccess 文件。

接下来，img 文件夹应该是空的，H5BP 原带的就是如此。

下面的 js 文件夹中应该包含以下子文件夹和文件：



由于采用了模块化的开发方式，Bootstrap 的 less 文件夹中包含很多文件。下面几节在讲到编译它们的时候，我们会详细介绍。

首先，我们来构造出 HTML 文件。

1.6　构造 HTML 模板

在新项目文件夹中，用编辑器打开 index.html。这个示例标记文件来自 H5BP，体现了一些最佳实践和建议方案。我们就以这个文件为基础，把它整合到 Bootstrap 的工作流中。下面先了解一下这个文件。

浏览一下整个文件，你会发现几个有意思的地方。这些地方在 H5BP 的文档中都有详细说明，访问这个链接 http://h5bp.com 很容易找到。不过我们接下来也会简单介绍一些，我们按照次序来。

- ❑ HTML5 文档类型声明：

```
<!DOCTYPE html>
```

- ❑ 针对 IE 的条件注释，开发者通过嵌套的选择符可以加入对旧版本浏览器的修复代码：

```
<!--[if lt IE 7]><html class="no-js lt-ie9 lt-ie8 lt-ie7"><![endif]-->
<!--[if IE 7]><html class="no-js lt-ie9 lt-ie8"><![endif]-->
```

```
<!--[if IE 8]><html class="no-js lt-ie9"><![endif]-->
<!--[if gt IE 8]><!--><html class="no-js"><!--<![endif]-->
```

- `html` 标签也包含一个 `no-js` 类。如果浏览器的 JavaScript 可用，Modernizr 脚本（本章前面介绍过）会把这个类删除，并将其替换成 `js` 类。如果这个类没有被删除，则表明 JavaScript 不可用，我们就需要使用嵌套的选择符为这种情况写一些 CSS 规则。
- 接下来是几个 `meta` 标签。
  - 用于指定字符集的：

    ```
    <meta charset="utf-8">
    ```

  - 告诉 IE 使用最新版的渲染引擎，或者如果安装了的话，使用谷歌的 Chrome Frame：

    ```
    <meta http-equiv="X-UA-Compatible"
      content="IE=edge,chrome=1">
    ```

  - 预留给描述站点用的：

    ```
    <meta name="description" content="">
    ```

  - 针对移动浏览器的视口标签：

    ```
    <meta name="viewport" content="width=device-width">
    ```

- 在该放站点图标和触摸屏图标的地方，是一行注释，告诉我们可以直接使用站点根目录下的图标文件，放在站点根目录下可以自动被用户浏览器及设备发现。
- 接下来是两个样式表的链接，一个指向 normalize.css，另一个指向 main.css：

```
<link rel="stylesheet" href="css/normalize.css">
<link rel="stylesheet" href="css/main.css">
```

- 再下面就是加载 Modernizr 脚本的 `script` 标签。这个脚本会为 IE8 提供 HTML5“垫片脚本”，以便它能识别 HTML5 的分区元素：

```
<script src="js/vendor/modernizr-2.6.2.min.js"></script>
```

- 接下来是 IE 条件注释，包含推荐用户把旧版 IE 升级到新版本的消息：

```
<!--[if lt IE 7]>
  <p class="chromeframe">You are using an
    <strong>outdated</strong> browser. ...
<![endif]-->
```

- 紧接着是一段文本。
- 随后是托管在谷歌服务器上的 jQuery 链接，以及一个本地 jQuery 的后备链接：

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/
  jquery.min.js"></script>
<script>window.jQuery || document.write('<script
  src="js/vendor/jquery-1.10.2.min.js">
  <\/script>')</script>
```

- 下面就是 plugins.js 和 main.js 的链接，分别用于保存 JavaScript 插件代码和我们编写的代码：

```
<script src="js/plugins.js"></script>
<script src="js/main.js"></script>
```

- 谷歌的 Analytics 脚本：

```
<script>
  var _gaq=[['_setAccount','UA-XXXXX-
    X'],['_trackPageview']];
  (function(d,t){var g=d.createElement(t),
     s=d.getElementsByTagName(t)[0];
  g.src=('https:'==location.protocol?'//ssl':'//www')+'
    .google-analytics.com/ga.js';
  s.parentNode.insertBefore(g,s)}(document,'script'));
</script>
```

如果你想更详细地了解有关上面这些内容的信息，还是去看看 H5BP 的文档吧，干脆直接把它的 HTML 文档链接给你：https://github.com/h5bp/html5-boilerplate/blob/v4.3.0/doc/html.md。这个文档中的解释非常详细。

对本章的任务而言，我们需要对这个模板中的元素进行如下操作：

(1) 设定我们站点的标题，针对旧版本浏览器用户更新现有的 IE 条件注释；
(2) 基于 LESS 文件编译 Bootstrap 的 CSS，添加基本的页面内容；
(3) 整合 Bootstrap 的 JavaScript 插件，确保响应式的导航条（`navbar`）正常响应。

做完这几件事之后，我们就可以真正开始设计自己的网站了。

1.7　设定站点标题

先把index.html文件中`<title>`的内容加上。你可以随便给自己的作品起名字，比如 Bootstrappin' Portfolio。为准确起见，这里使用HTML实体`&#39`来表示单引号，结果如下所示：

```
<title>Bootstrappin&#39; Portfolio</title>
```

1.7.1　调整过时的浏览器消息

模板中的消息针对老浏览器用户。大概在第 20 行左右，消息内容如下：

```
You are using an outdated browser. Please upgrade your browser
or activate Google Chrome Frame to improve your experience.
```

这段消息里面包含一个指向 http://browsehappy.com 的链接，该网站是一个推荐浏览器升级的站点，也包括升级到让 IE 能拥有现代浏览器能力的插件——Google Chrome Frame。

（不过，随着谷歌从 2014 年 1 月起停止维护，Google Chrome Frame 的链接也许已经不存在了。）

在本书写作的时候，这条消息包含在一个只针对 IE7 之前浏览器（即 IE6、IE5，等等）的条件注释中。这样一来，除非看源代码，否则不太可能有人看到这条消息。

世界一直在前进。很多组织都在升级浏览器，而很多设计者也不再支持 IE7。不过，一般来说，大家的目标还是要确保 IE7 用户能够看到站点内容，只是不保证他们的体验。

这样做是比较实际的。因为要完全支持 IE7，必须增加很多额外的工作量，包括 CSS 和 JavaScript 编码工作。代码增加，带宽占用也增加，成本也更高。

所以，Bootstrap 3 也不再支持 IE7。但在开发结束后，我们还是应该测试一下，保证 IE7 用户能够访问站点。当然，肯定不能保证完美的体验了。

为此，我们应该让 IE7 用户也看到这条消息。这就需要在原有条件注释中添加一个表示等于的 `e`（equal），如下所示：

```
<!--[if lte IE 7]>
```

也就是把原来的 `lt` 替换成 `lte`。

还有几点要注意。

> 对于 IE7 及更早的 IE 版本，可以考虑提供基本的样式表，保证用户能够使用你的网站。
>
> 如果你的用户里有很大一部分都使用 IE7，而且这些人也不可能升级到新版本，可以考虑支持 IE7 的 Bootstrap 2.2.3。

要是你想知道这些消息显示在浏览器里是什么样子，或者想给它们添加一点样式，可以暂时把开始（`<!--[if lte IE 7]>`）和结束（`<![endif]-->`）的条件注释删掉或注释掉。

1.7.2　设置主结构元素

下面开始准备页面内容。目前，还只有一个段落。我们可以稍微添加一点东西，比如下列内容：

- 包含 Logo 和导航的页头区；
- 包含页面内容的内容区；
- 包含版权和社交媒体链接的页脚区。

添加这些内容，我们都会基于最新的 HTML 最佳实践来做，而且会考虑 ARIA（Accessible Rich Internet Applications，可访问富因特网应用）的 `role` 属性（即 `banner`、`navigation`、`main` 和 `contentinfo` 这几个角色）。可能你也知道，HTML5 后来又增加了`<main role=`

"main"></main>元素，目的是专门为页面或分区中的主内容提供一个专用的元素。要了解更多相关信息，请参考这个链接：http://www.sitepoint.com/html5-main-element/。

找到模板文件中下面这个注释和段落：

```
<!-- Add your site or application content here -->
<p>Hello world! This is HTML5 Boilerplate.</p>
```

将它们替换成下面这样：

```
<header role="banner">
  <nav role="navigation">
  </nav>
</header>

<main role="main">
  <h1>Main Heading</h1>
  <p>Content specific to this page goes here.</p>
</main>

<footer role="contentinfo">
  <p><small>Copyright &copy; Company Name</small></p>
</footer>
```

这就是我们页面的基本结构和内容了。接下来更进一步。

1.8　导航条

还记得吧，咱们并没有借用 Bootstrap 预编译的 CSS 文件，而且也没有自己写 LESS 并编译自己的 CSS；稍后我们会。在此之前，我们得先把 Bootstrap 特有的元素设置好，那就是导航条。

作为起点，我们可以就使用 Bootstrap 基本的导航条（后面再添加更多细节）。为此，我从 Bootstrap 文档中拿来它的导航条代码，然后做了如下调整：

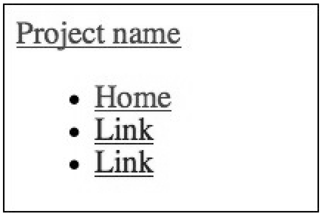
- 添加了 navbar-static-top 类，因为我们希望导航条能够定位到窗口顶部，但能够随页面滚动而滚动；
- 把项目名称链接到 index.html；
- 把原来的父 div 标签改成了语义化的 HTML5 nav 标签。

经过这一番调整后，得到如下 header 元素：

```
<header role="banner">
  <nav role="navigation" class="navbar navbar-static-top navbar-default">
    <div class="container">
      <div class="navbar-header">
        <a class="navbar-brand" href="index.html">Project name</a>
      </div>
      <ul class="nav navbar-nav">
```

```
        <li class="active"><a href="index.html">Home</a></li>
        <li><a href="#">Link</a></li>
        <li><a href="#">Link</a></li>
      </ul>
    </div>
  </nav>
</header>
```

保存结果，在浏览器中打开并刷新 index.html。如下图所示，并没有太多特别的：



内容有了。现在，我们特别需要自己的样式表。先来编译并链接 Bootstrap 默认的样式表。

1.9 编译和链接默认的 Bootstrap CSS

我们可以直接把 Bootstrap 默认的 bootstrap.css 文件拿来用，但不如借此机会学习一下编译 LESS 文件。这样可以为我们将来的设计打下基础。

1.9.1 编译 Bootstrap CSS

可能大家有熟悉 LESS，也有不熟悉 LESS 的。没关系，不管你是否熟悉，我都会教你怎么做。不过，我还是建议你看看 LESS 的文档：http://lesscss.org，再找几个 LESS 的教程看完。稍后你就会知道，LESS 非常强大，也很好玩，使用它能够大大提高效率。

现在，我们只编译，而不写 LESS 文件。

找到 less/bootstrap.less，在编辑器中打开它。你会发现这个文件导入了 less 文件夹中所有的其他文件。编译后，这个文件会生成完整的 bootstrap.css 样式表。而这就是我们第一步要做的。

如果你以前没有编译过 LESS 文件，需要下载和安装它的编译器。

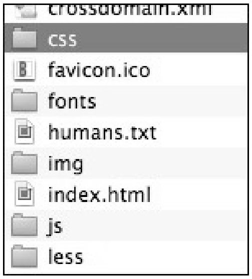
- Windows 用户，下载安装这个编译器：
  - WinLess（免费桌面应用），地址为 http://winless.org[①]。

① 为方便起见，读者也可以在这个链接下载：http://pan.baidu.com/s/1c03tWQW。——译者注

- Mac 用户，可以选择下载：
  - Crunch 应用（免费），地址为 http://crunchapp.net/。
  - CodeKit（收费），地址为 http://incident57.com/codekit/。

下载了选择的 LESS 编译器之后，安装，打开。然后就可以按照下面的步骤来做了。

(1) 在 fonts、img、js 和 less 文件夹平级的主文件夹里创建一个 css 文件夹。



(2) 使用下列方法中的一种把主文件夹（css、fonts、img、js 和 less 文件夹的父文件夹）添加到编译器：

- 把文件夹拖到编译器窗口中；
- 在编译器窗口中找到 Add folder 按钮，点击后选择主文件夹。

(3) 然后在编译器窗口中可以看到加载的 LESS 文件，找到 less/bootstrap.less 文件。

(4) 右键单击 less/bootstrap.less 文件，选择 Select output file，找到刚刚创建的 css 文件夹，此时输出文件名应该自动会变成 bootstrap.css，单击“保存”。

(5) 选择输出路径和文件名，单击 Compile。

(6) css 文件夹中会出现编译生成的 bootstrap.css 文件。

> 如果编译出了问题，可以查看编译器的日志，以及输出路径是否正确。此外，如果编译时出错，也可能是编译器的更新没有跟上 LESS 开发的步伐。最近，我在使用另外一个免费编译器时就碰到了这样的问题。如果你发现编译器不能编译默认的 Bootstrap LESS 文件，很可能需要下载编译器的最新版本，或者编译器本身需要更新了。

(7) 编译成功后，唯一要注意的是这个文件名是否与 index.html 中链接的文件名相同。

(8) 在 index.html 中，删除指向 css/normalize.css 的样式表链接，因为这个样式表已经包含在 Bootstrap 中了（normalize.less 在 bootstrap.less 中是第一个导入的）。

(9) 接下来链接的样式表指向 css/main.css，因为后面我们会自定义 Bootstrap 以生成自己的样式表，所以这个链接先不用动，将来我们再用它来链接自定义的样式表。

(10) 在这里，我们先来个偷梁换柱，复制一份 bootstrap.css，重命名为 main.css（将来再用自定义的样式表重写这个文件），结果如下图所示：



(11) 刷新浏览器，应该看到 Bootstrap 3 默认的导航样式，如下图所示，从排版和布局上有所增强，这说明 CSS 已经生效，祝贺你！



这样我们就配置好了使用 Bootstrap 的默认样式了。接下来，我们就继续把导航条变成响应式的。在此期间，我们还会顺便测试 Bootstrap 的 JavaScript 插件能够正常工作。

1.9.2 完成响应式导航条

为了在 Bootstrap 响应式导航条基础上完成我们的导航条，还得再增加两个新元素，以及相应的类和 `data` 属性。相关的用法可以参考 Bootstrap 的 Components 文档，在 Navbar 选项卡下：http://getbootstrap.com/components/#navbar。

先按照下列步骤添加额外的标记。

(1) 搜索到`<div class="navbar-header">`,在这个元素中,添加一个 `navbar-toggle` 按钮，用于展开和收起响应式导航条。下面就是这个按钮的全部标记（我把它放到 `navbar-header` 里面）：

```
<div class="navbar-header">
  <button type="button" class="navbar-toggle" data-
    toggle="collapse" data-target=".navbar-collapse">
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
  </button>
  <a class="navbar-brand" href="index.html">Project
    name</a>
</div>
```

下面简单解释一下以上代码：

- 按钮中的 `navbar-toggle` 类用于应用 CSS 样式；
- 后面的数据属性 `data-toggle` 和 `data-target` 是 Bootstrap 的 JavaScript 插件要用的，分别表示预期行为和预期目标（即 `collapse` 和类名为 `navbar-collapse` 的元素，这个元素后面会添加）；
- 类名为 `icon-bar` 的 span 元素是 CSS 用来创建按钮中的三道杠按钮用的。

(2) 接下来把导航项包装在一个收起的 `div` 中，即用带有适当 Bootstrap 类的 `div` 把 `ul class="nav navbar-nav">`包装起来：

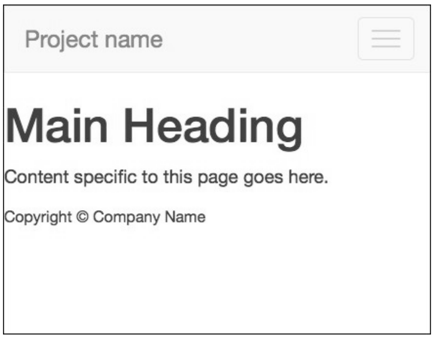
```
<div class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
    <li class="active"><a href="index.html">Home</a></li>
    <li><a href="#">Link</a></li>
    <li><a href="#">Link</a></li>
  </ul>
</div><!--/.nav-collapse -->
```

在前面两步中，我们把代码分隔成两部分，而且都位于`<div class="container">`中。为确保你的代码写得没错，可参考本章完整的示例代码。

> 这里的标签名、类名和数据属性在将来的 Bootstrap 版本中可能会变。如果你发现自己的代码不行，一定要看看相应 Bootstrap 版本的文档。保险起见，可以使用本书提供的示例代码来试验。

好了，保存文件，刷新浏览器。在任何一个现代浏览器（IE9 或 Firefox、Chrome、Safari 等的最新版本）中，拖动窗口缩小到小于 980 像素。

如果一切顺利，应该看到收起来的导航条，如下面的屏幕截图所示，但可以看到站点名或 Logo 以及切换按钮。



这是个好兆头！再单击切换按钮，应该看到滑动打开的链接，如下图所示：



成功啦，干得不错！

1.9.3　排除故障

如果一切顺利，那说明你已经成功地把 LESS 编译成了 CSS，而且也成功地包含了 Bootstrap 的 JavaScript 插件。

如果不顺利，可以再检查一遍以下事项。

- 你的标记结构嵌套关系是否正确？有没有未闭合、不完整或错配的标签 、类，等等？
- 是否成功地把 LESS 编译成了 CSS？编译得到的 CSS 是否放到了正确的文件夹里，命名是否正确？
- 位于 index.html 中的 CSS 链接是否正确？
- 是否包含了 Bootstrap 的 JavaScript 插件？

另外，下面这些事项也可能有用。

(1) 把前面的步骤从头到尾再过一遍，同时注意上述各项；
(2) 验证 HTML 保证格式正确；
(3) 比较本书示例代码和你自己的代码；
(4) 参考 Bootstrap 文档，看是否有标签结构和属性的变化；
(5) 把你的代码放到 http://jsfiddle.net/或 http://www.codepen.com/上，到 http://stackoverflow.com/上寻找高手帮助。

把那么多文件揉合到一起，让它们协作运行，出点问题很正常。而学会找到问题和解决问题同样是我们的生存之道！

好吧，假设到现在为止所有问题都解决了，接下来就让我们再探讨一个不那么显而易见的问题。我们想让自己的作品支持 IE8。为此，需要考虑一下老版本的浏览器。

1.9.4 支持 IE8

要支持 IE8，需要一段 JavaScript 代码让浏览器能够响应媒体查询。这段代码就是 Scott Jehl 的 respond.js "腻子脚本"。

Bootstrap 自身的文档推荐这样做以兼容 IE8。相关的信息可以参考这里：http://getbootstrap.com/getting-started/#browsers。

为了针对 IE8 应用这段脚本，需要针对 IE8 的条件注释：

```
<!--[if lt IE 9]>
...
<![endif]-->
```

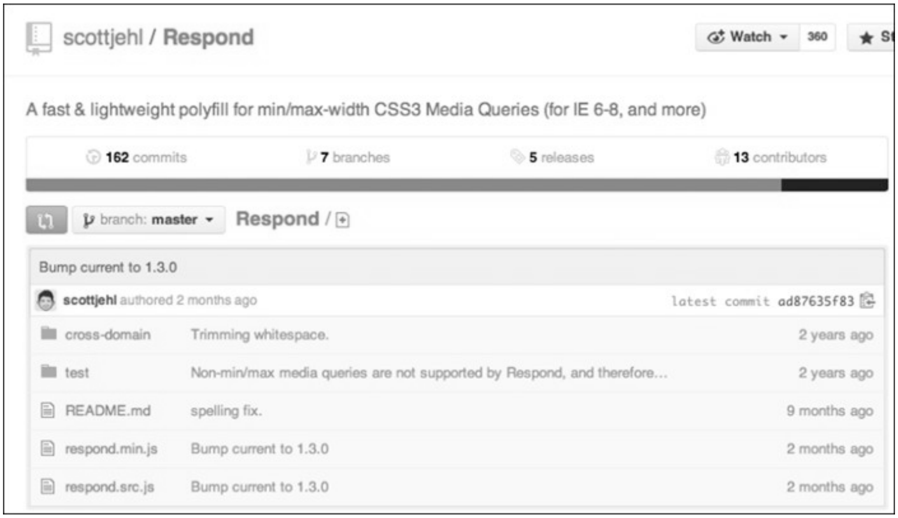
另外，根据 Andy Clarke 的建议，为了不让并不需要这个脚本的 Windows 移动设备加载该脚本，还应该排除 IE 移动版浏览器，具体参见他的在线代码库 320andup，地址是：https://github.com/malarkey/320andup/。

Clarke 建议的条件注释如下：

```
<!--[if (lt IE 9) & (!IEMobile)]>
...
<![endif]-->
```

有了条件注释，下面就是在站点模板文件中添加腻子脚本了，步骤如下。

(1) 打开 https://github.com/scottjehl/Respond（也可以在 Github 上搜索 respond.js 找到下载地址）。如果你有时间，可以看看这个页面中的文档，了解一些这个脚本的工作原理。

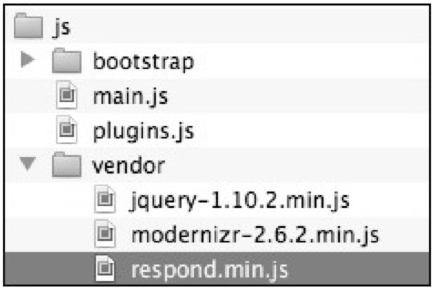


(2) 找到文件，下载 ZIP：



(3) 解压缩，找到名为 respond.min.js 的压缩版。

(4) 把它复制到项目文件夹中的 js/vendor 目录下，与 jQuery 和 Modernizr 放到一块。



(5) 然后，把下面几行加载 respond.js 文件的代码添加到 index.html 中，包括针对 IE 的条件注释，就在加载 Modernizr 的代码下面：

```
<!-- Modernizr -->
<script src="js/vendor/modernizr-2.6.2.min.js"></script>
<!-- Respond.js for IE 8 or less only -->
<!--[if (lt IE 9) & (!IEMobile)]>
  <script src="js/vendor/respond.min.js"></script>
<![endif]-->
```

(6) 好了，这样 IE8 就可以支持媒体查询响应视口大小变化了。

> 如果你想测试添加腻子脚本的结果，但又没有 IE8 浏览器，可以使用一个在线服务，叫 Browsershots，地址是：http://browsershots.org，这是免费的。还有一个收费的，叫 BrowserStack，地址是：http://www.browserstack.com（试用免费）。

这样，我们站点的模板就做完了。继续之前，我们先来小结一下。

1.10 小结

如果大家一直跟着前面的教程在做，那到现在已经为继续实现更高级的设计打好了基础。我们来清点一下吧，我们已经有了：

- ❑ 一个完善的 HTML5 标记结构，内置了很多最佳实践；
- ❑ 一个标准的 Bootstrap 样式表文件，已链接；
- ❑ 能够正常工作的 JavaScript 插件；
- ❑ 一个响应式的导航条；
- ❑ （可能更重要的）随时可以派上用场的 LESS 编译器。

> 这时候，或许把所有文件都备份一下比较好，因为后面的项目都可以把它作为基础。

下一章我们就来弄点好玩的，进一步挖掘 Bootstrap 的潜力，创建一个漂亮的个人作品展示站点。

作品展示站点

2

假设我们已经想好了要给自己的作品弄一个在线站点。一如既往，时间紧迫。我们需要快一点，但作品展示效果又必须专业。当然，站点还得是响应式的，能够在各种设备上正常浏览，因为这是我们向目标客户推销时的卖点。这个项目可以利用 Bootstrap 的很多内置特性，同时也将根据需要对 Bootstrap 进行一些定制。

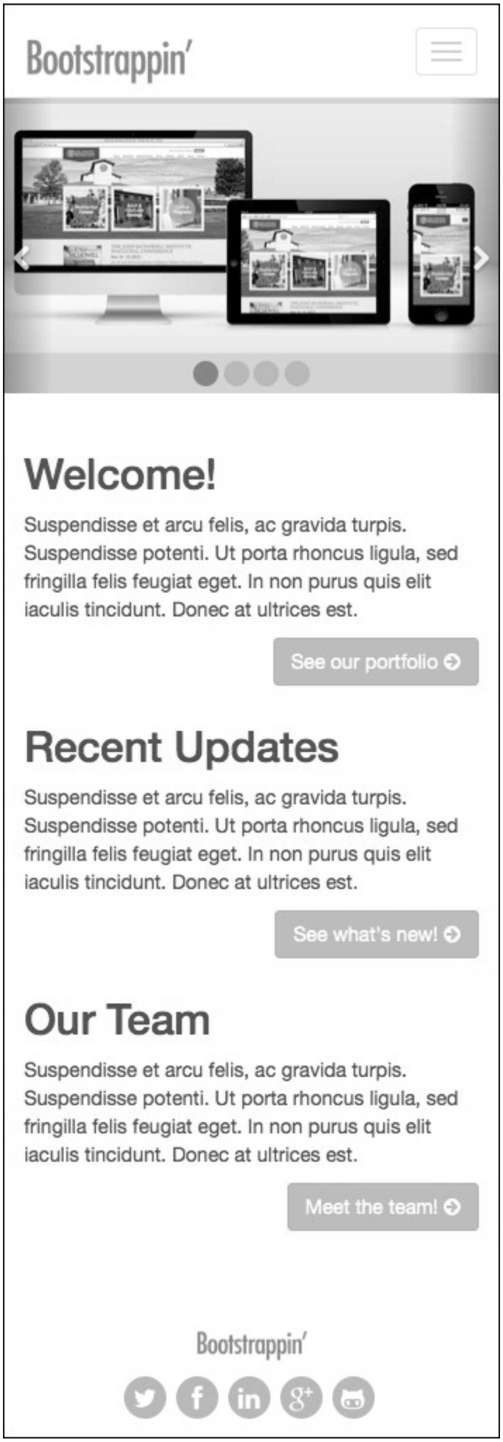
2.1 设计目标

我们已经草拟了两个主页的效果图。虽然对大屏幕中的展示效果已经胸有成竹，但我们还应该从小屏幕设备开始，强迫自己聚焦在关键的要素上面。

这个作品展示站点应该具有下列功能：

- 带 Logo 的可折叠的响应式导航条；
- 重点展示 4 张作品的图片传送带；
- 单栏布局中包含三块内容，每块内容中都包含标题、短段落和吸引人点击阅读的大按钮；
- 页脚包含社交媒体链接。

下面的屏幕截图展示了设计方案：


Bootstrappin'
Welcome!
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
See our portfolio
Recent Updates
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
See what's new!
Our Team
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Meet the team!
Bootstrappin'

总的来看，这将是对我们工作成果的一个完美的展示。图片传送带比较高，可以充分展示我们作品的图片。当然，导航到底下的内容也不难，用户可以先了解每一项的大致情况，然后决定深入阅读哪块内容。通过把重要的链接做成大按钮，从视觉上突出了重要的操作，可以起到吸引用户点击的作用，而且就算是手指粗大的用户都可以轻易点触。

为了便于维护，我们决定只考虑两个主要的断点。在小于 768px 的小屏幕中使用单栏布局，否则就切换到一个三栏布局：



在这个针对大屏幕的设计效果图中，可以发现下列功能：

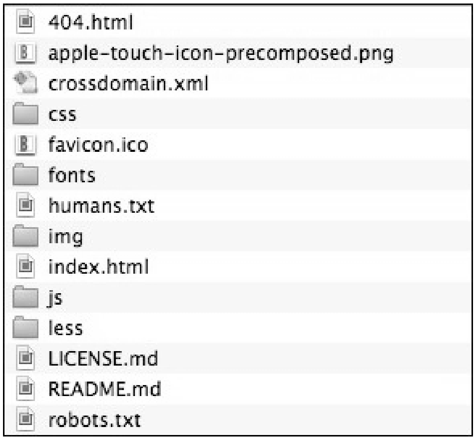
- 位于顶部的导航条，而且各导航项都附带图标；
- 宽屏版的图片传送带，其中的图片拉伸至与浏览器窗口同宽；
- 三栏布局分别容纳三块文本内容；
- 页脚在布局中水平居中。

这个设计的配色很简单，只有灰阶和用于链接和突出显示的金绿色。

明确了设计目标，接下来就可以布置内容了。

2.2　查看练习文件

我们来看看这个练习用的文件，就在本书源代码的 02_Code_BEGIN 文件夹中。看起来这里的文件与第 1 章中模板文件夹差不多。

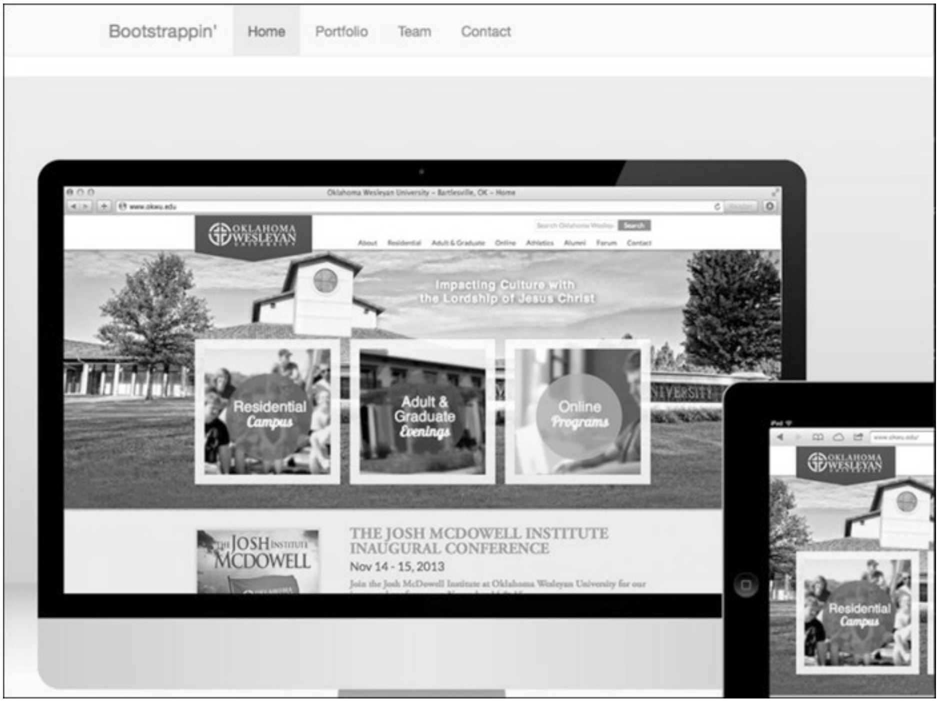


新增的一些内容如下。

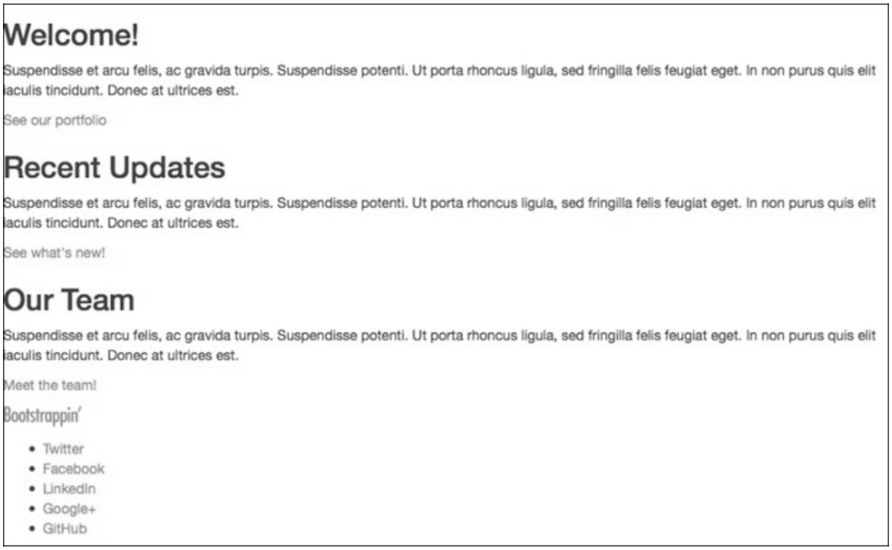
- less 文件夹里有一些改动，但我们不着急介绍，还是先看看与内容与关的变化吧。
- img 文件夹现在包含 5 张图片：
  - 1 张 Logo 图片，名为 logo.png；
  - 4 张作品图片。
- index.html 的改动之处如下：
  - 导航栏更新了，以反映新站点的架构；
  - 图片、内容块、页脚中 Logo 和社交链接都有了各自基本的标记。

与第 1 章中的导航栏不同，此时的传送带、分栏和图片都没有添加 Bootstrap 类。可以在浏览器中看一看结果。

打开页面后，可以看到导航栏后面就是作品图片：



作品图片后面就是文本块和包含一组社交链接的页脚：



倒也简单。还是开始让它变身吧！

我们从添加 Bootstrap 类着手，这样可以利用 Bootstrap 默认的 CSS 样式和 JavaScript 行为，迅速搭建起基本的界面元素。

2.3　搭建传送带

下面先来搭建传送带，传送带会循环展示我们作品的 4 张特写图片。

Bootstrap 传送带的标记结构可以在其文档页面找到，URL 为：http://getbootstrap.com/javascript/#carousel。

可以按照示例中的结构设置其中的元素。以下代码就是传送带的所有标记，包含各个部分，首先是进度指示器：

```
<div id="homepage-feature" class="carousel slide">
  <ol class="carousel-indicators">
    <li data-target="#homepage-feature" data-slide-to="0"
      class="active"></li>
    <li data-target="#homepage-feature" data-slide-to="1"></li>
    <li data-target="#homepage-feature" data-slide-to="2"></li>
    <li data-target="#homepage-feature" data-slide-to="3"></li>
  </ol>
```

整个传送带是一个带 ID 属性（`id="homepage-feature"`）的 `div` 标签，其 `carousel` 类用于把 Bootstrap 的传送带 CSS 应用到这个元素，为指示器、传送带项和前一张及后一张控件添加样式。

进度指示器的 `data-target` 属性必须使用传送带的 ID `homepage-feature`。有了这个属性，JavaScript 插件才能为活动的传送带项添加 `active` 类。在此我们预先为第一个指示器添加了 `active` 类。然后，类的切换就由 JavaScript 控制了。它会删除第一个指示器的这个类，再添加到后续指示器，如此循环。

此外，还要注意 `data-slide-to` 的值从 0 开始，与 JavaScript 和其他编程语言一样。记住：从 0 开始，不是从 1 开始。

指示器后面，是类为 `carousel-inner` 的元素。这个元素是所有传送带项（item），也就是所有图片。

`carousel-inner` 元素内部是传送带项，是一组 `div` 标签，每个都带着 `class="item"`。把第一项修改成包含 `item` 和 `active` 两个类，使其一开始就可见。

此时的标记结构如下所示：

```
<!-- Wrapper for slides -->
<div class="carousel-inner">
  <div class="item active">
    <img src="img/okwu-athletics.jpg" alt="OKWU Athletics Homepage">
  </div>
```

```
    <div class="item">
      <img src="img/okwu.jpg" alt="OKWU.edu Homepage">
    </div>
    <div class="item">
      <img src="img/bso.jpg" alt="Bartlesville Symphony Homepage">
    </div>
    <div class="item">
      <img src="img/alittlecode.jpg" alt="aLittleCode.com Homepage">
    </div>
  </div><!-- /.carousel-inner -->
```


传送带项之后，还需要添加传送带控件，用于在传送带左、右两侧显示前一个和后一个按钮。你会发现其中有类对应着 Glyphicon 字体图标。在控件后面，我们再用一个结束 div 标签关闭整个传送带。

```
<!-- Controls -->
  <a class="left carousel-control" href="#homepage-feature" data-slide="prev">
    <span class="glyphicon glyphicon-chevron-left"></span>
  </a>
  <a class="right carousel-control" href="#homepage-feature" data-slide="next">
  <span class="glyphicon glyphicon-chevron-right"></span>
  </a>
</div><!-- /#homepage-feature.carousel -->
```

> 每个传送带控件（carousel-controls）的 href 属性必须是最外围传送带元素的 ID 值（#homepage-feature）。

添加以上代码之后，请保存并刷新浏览器。Bootstrap 的样式和 JavaScript 都应该生效，页面中的图片应该变身成为滚动的传送带！

默认情况下，传送带的幻灯片每 5 秒切换一次。为了充分展示我们的作品，可以改成 8 秒。

(1) 打开 js/main.js

(2) 添加以下代码。这里先用 jQuery 方法检测相应的页面元素是否存在，如果存在则将传送带的间隔时间初始化为 8000 毫秒。

```
$( document ).ready(function() {
  $('.carousel').carousel({
    interval: 8000
  });
});
```

(3) 保存并刷新。你会看到间隔时间加长到了 8 秒。

相关的选项可以参考 Bootstrap 传送带的文档：http://getbootstrap.com/javascript/#carousel。

关于定制传送带及其指示器和图标的样式，本章后面再讨论。我们接下来继续看如何利用 Bootstrap 默认的样式，为传送带下面的内容块设置响应式网格。

2.4　创建响应式分栏

页面中有三块文本，每块都有标题、段落和链接。在大于等于平板电脑的屏幕上， 我们希望内容分三栏，而在较窄的屏幕上，我们希望内容变成一栏全宽。

建议大家花点时间熟悉一下 Bootstrap 移动优先的响应式网格，文档地址是：http://getbootstrap.com/css/#grid。

简单地说，Bootstrap 内置 12 栏网格系统，其基本的类结构支持 col-12 表示全宽，col-6 表示半宽，col-4 表示三分之一宽，以此类推。

在 Bootstrap 3 中，由于创造性地使用了媒体查询，网格系统具有极强的适应力。如前所述，我们希望欢迎消息在比平板小的屏幕中呈现为一栏全宽，而在大约 768px 时变成三分之一栏宽。巧合的是，Bootstrap 内置的小屏幕断点恰好是 768px，也就是 @screen-sm-min 变量的默认值。而大于 768px 的中屏幕断点是 992px，对应变量是 @screen-md-min。然后，大于 1200px 断点的算大屏幕。这几个断点我们后面统称为小、中、大断点。

小断点有一个特殊的栏类命名法：col-sm-。因为我们想在小断点之上使用三栏，所以这里使用 class="col-sm-4"。这样在小断点之下，分块元素会保持全宽，而在小断点之上，则会各占三分之一宽并肩排列。完整的结构如下所示，为简明起见，段落内容作了省略处理：

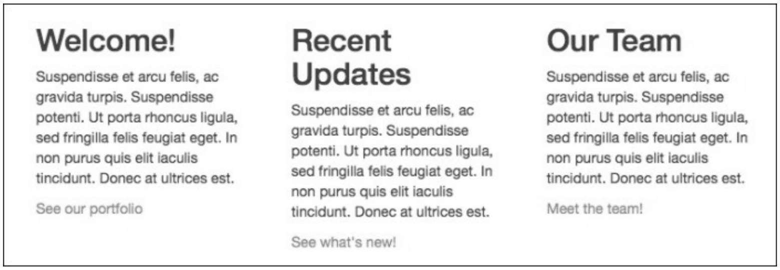
```
<div class="container">
  <div class="row">
    <div class="col-sm-4">
    <h2>Welcome!</h2>
    <p>Suspendisse et arcu felis ...</p>
    <p><a href="#">See our portfolio</a></p>
  </div>
  <div class="col-sm-4">
    <h2>Recent Updates</h2>
    <p>Suspendisse et arcu felis ...</p>
    <p><a href="#">See what's new!</a></p>
  </div>
  <div class="col-sm-4">
    <h2>Our Team</h2>
    <p>Suspendisse et arcu felis ...</p>
    <p><a href="#">Meet the team!</a></p>
  </div>
  </div><!-- /.row -->
</div><!-- /.container -->
```

下面解释一下 container 和 row 类的作用：

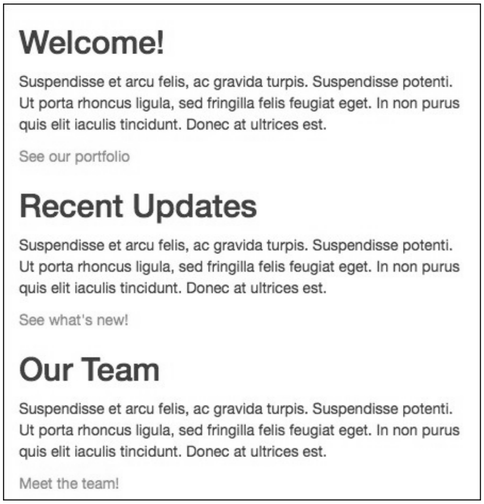
- container 类用于约束内容的宽度，并使其在页面内居中；
- row 类用于包装三个栏，并为栏间留出左右外边距；

- container 类和 row 类都设定了清除，因而它们可以包含浮动元素，同时又清除之前的浮动元素。

现在，保存并刷新。在浏览器窗口宽度超过 768px 时，应该看到下图所示的三栏布局：



把窗口缩小到 768px 以下，又会看到三栏变成了一栏：



好，这样就利用响应式网格系统完成了响应式分栏，接下来我们要利用 Bootstrap 的按钮样式，把内容分块中的链接做成突出的效果。

2.5 把链接变成按钮

把重要的内容链接转换成突出显示的按钮很简单。为此要用到如下几个关键的类：

- btn 类用于把链接变成按钮的样式；
- btn-primary 类用于把按钮变成主品牌颜色；
- pull-right 类用于把链接浮动到右侧，使其占据更大的空间，从而更便于发现和点击。

把上述这几个类添加到三个内容块末尾的链接上：

```
<p><a class="btn btn-primary pull-right" href="#">See our portfolio</a></p>
```

保存并刷新，应该能够看到类似下图所示的结果：



我们又前进了一大步。关键的内容元素已经基本成型了。

在基本的标记结构就位的条件下，接下来可以进行微调了。为此需要用到自定义的 CSS，而我们要借此机会体验一下 Bootstrap 的 LESS 文件的强大威力。不熟悉 LESS 也不必担心，我会一步一步教你怎么做。

2.6　理解 LESS

本节我们会学习创建、编辑、定制一些 LESS 文件，以便为我们的设计生成期望的 CSS。

> 如果你不太了解 LESS，希望进一步学习，可以参考以下链接。
> - LESS 文档：http://lesscss.org/#docs；
> - Sitepoint 网站关于 LESS 的完整介绍：http://www.sitepoint.com/a-comprehensive-introduction-to-less/。

简言之，使用 LESS 预处理器来生成 CSS 是一件既令人激动又十分轻松的事。接下来我们就具体讨论。

2.6.1　嵌套规则

嵌套规则极大提高了组合样式的效率。比如，CSS 中的选择符可能会多次重复出现：

```
.navbar-nav { ... }
.navbar-nav > li { ... }
.navbar-nav > li > a { ... }
.navbar-nav > li > a:hover,
.navbar-nav > li > a:focus { ... }
```

其中这些相同的选择符用 LESS 写会简洁很多，只要使用一个简单的嵌套即可：

```
.navbar-nav { ...
  > li { ...
    > a { ...
      &:hover,
      &:focus { ... }
    }
  }
}
```

编译后，这些规则会生成标准的 CSS。但 LESS 的嵌套规则更容易写，也更容易维护。

2.6.2 变量

使用变量可以让我们只设定（或修改）一次，就能自动影响（更新）整个样式表中用到该值的属性。比如，可以像下面这样使用颜色变量：

```
@off-white:  #e5e5e5;
@brand-primary:  #890000;
```

在这些变量的值变化后，可以自动将它们更新到整个站点。这是因为我们在 LESS 文件中使用了这些变量：

```
a {
  color: @brand-primary;
}
.navbar {
  background-color: @brand-primary;
  > li > a {
  color: @off-white;
  }
}
```

2.6.3 混入

混入可以让生成整套规则更方便也更容易管理。比如，可以利用它简化为元素应用 `border-box` 属性的任务。在 CSS 中，要涵盖所有浏览器，需要用到每种浏览器的供应商前缀，为此针对每个元素都要写三行相同的声明，而且还要记住每一种前缀的写法：

```
.box {
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
}
```

在 LESS 中，可以只写一个供混入的规则，后面则可以通过`@boxmodel` 参数来指定期望的盒模型：

```
.box-sizing(@boxmodel) {
  -webkit-box-sizing: @boxmodel;
     -moz-box-sizing: @boxmodel;
          box-sizing: @boxmodel;
}
```

然后就可以在需要的地方使用这个混入了：

```
.box {
  .box-sizing(border-box);
}
.another-element {
  .box-sizing(border-box);
}
```

编译之后，还会为每个元素生成三行 CSS。

2.6.4　运算式

通过运算式可以基于变量实现数学计算。比如，可以将一种颜色作为基准，对其进行加亮和减暗处理：

```
a:hover { darken(@link-color, 15%); }
```

还可以计算内边距的值，以适应导航条的高度。比如，以下 Bootstrap 的 navbar.less 文件中的代码，就将导航项的内边距值设定为导航条高度减去行高之后剩余的高度值。然后，把这个值一分为二，平均应用为顶部和底部内边距：

```
.navbar > li > a {
  padding-top: ((@navbar-height - @line-height-computed) / 2);
  padding-bottom: ((@navbar-height - @line-height-computed) / 2);
}
```

2.6.5　导入文件

LESS 编译器支持导入并组合多个文件，最终生成一个统一的 CSS 文件。我们可以指定导入的次序，按照需要的层叠关系精确组织结果样式表。

Bootstrap 的导入文件 bootstrap.less 一开始导入了基本的变量和混入。然后，又导入了（代替 CSS 重置样式表的）normalize.css 的 LESS 版，之后是针对打印媒体的基本样式（print.less）、基本的全局样式（scaffolding.less）和排版样式（type.less）。结果这个 bootstrap.less 文件的开头几行就是这样的：

```
// Core variables and mixins
@import "variables.less";
@import "mixins.less";

// Reset
```

```
@import "normalize.less";
@import "print.less";

// Core CSS
@import "scaffolding.less";
@import "type.less";
```

而最终得到的将是一个统一的 CSS 文件，其中的样式经过从一般到特殊的层叠，正是我们想要的。

2.6.6　模块化

把多个文件导入并输出成一个文件，可以让我们更方便地组织样式，对它们进行分组，并在不同的文件中对它们进行维护。这也是 Bootstrap 会带有那么多 LESS 文件的原因，导航条有一个，按钮有一个，警告框有一个，传送带有一个……，而所有这些都会被导入到 bootstrap.less 文件。

正因为上述原因，LESS 及其他 CSS 预编译器才会变得如此流行。它们已经成为专业 Web 开发不可或缺的一个重要部分。多数开发人员都认为这正是 CSS 的发展方向。

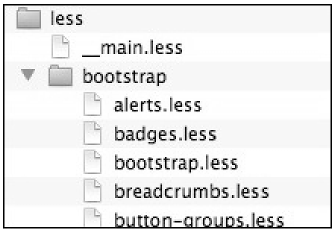
2.7　根据需要定制 Bootstrap 的 LESS 文件

在定制 Bootstrap 的 LESS 文件期间，我们会发挥很大的主观能动性，具体如下：

- 组织 less 文件夹，以便灵活、自由地实现我们需要，同时也让将来的维护者更方便；
- 自定义 Bootstrap 提供的几个 LESS 文件；
- 创建几个新的 LESS 文件；
- 为站点整合一套较大的字体图标，数量翻倍，并将图标运用于社交媒体链接。

换句话说，我们可不光是要学习应用 Bootstrap 的约定，还要发挥自己的创造力。

在本章的练习文件中，打开 less 文件夹。打开后，可以看到下图所示的目录结构：



为方便起见，我已经把 Bootstrap 的 LESS 文件提前集中到了 bootstrap 子文件夹下面。

文件__main.less 是 bootstrap.less 的重命名版，它导入了其他所有文件，将来就是通过编译

它来基于所有导入的 LESS 文件生成一个统一的样式表。打开__main.less 后，你会发现它绝大部分内容与 bootstrap.less 一样，只不过导入文件的路径中多了一个 bootstrap 文件夹。

```
// Core variables and mixins
@import "bootstrap/variables.less";
@import "bootstrap/mixins.less";
```

为什么要多此一举呢？因为我们很快就要创建自己的 LESS 文件了。这样一来，我们创建的文件就不会与 Bootstrap 内置的文件混淆，便于调整。

或许有读者会问，为什么这个文件名开头要使用两个下划线？事实上，原因有四。

- 按字母排序时，下划线可以让文件排在前头。
- 它不是我们唯一的自定义文件，如果其他自定义文件只使用一个下划线，那么它就显得更突出。
- 采用这种命名方式，更便于我们扫描或搜索自定义文件。由于下划线比较显眼，所以在搜索时，只输入一个下划线，就可以列出所有自定义文件。
- 同时打开多个文件编辑时，带下划线的文件名也容易让我们找到相应的标签页。

优点不少吧，下面开始自定义！先从自定义 Bootstrap 变量和添加新变量开始。

自定义变量

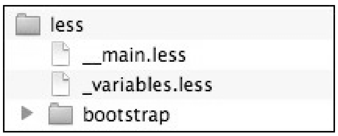
按照惯例，我们先复制一份 Bootstrap 的变量文件，然后对其进行修改。

(1) 找到 less/bootstrap 文件夹中的 variables.less 文件，在编辑器中打开它。

(2) 浏览一下这个文件，会发现各种变量，有定义基本颜色值的，有定义页面背景颜色的，有定义字体的，还有定义导航条和背景高度的，等等。看起来很好，但改动一下更妙。改动之前，我们先另存一个副本，以防将来改坏，好恢复。

(3) 把副本另存到 bootstrap 文件夹外部，在 less 文件夹的__main.less 文件旁边。为表示它是自定义文件，在原文件夹前添加一个下划线，变成_variables.less。

现在，less 文件夹的截图如下所示：



下面我们就来自定义颜色。

(1) 在新_variables.less 文件的最开始，可以看到 Bootstrap 为灰色和品牌色定义的默认变量及其值。这里的灰色值是基于黑色按比例计算的，使用了 LESS 的 `lighten` 函数：

```
@gray-darker:             lighten(#000, 13.5%); // #222
@gray-dark
```

(2) 我们知道自己想要的值，因此直接替换即可（你也可以尝试一下使用计算的值）。然后再增加两个变量，以涵盖我们需要的完整灰度空间。

(3) 结果如下：

```
@gray-darker:           #222;
@gray-dark:             #454545;
@gray:                  #777;
@gray-light:            #aeaeae;
@gray-lighter:          #ccc;
@gray-lightest:         #ededed;
@off-white:             #fafafa;
```

接下来再更新品牌颜色中的`@brand-primary`变量，将其改为金黄色：

```
// Brand colors
// -------------------------
@brand-primary:         #c1ba62;
```

要看结果，需要导入这些新变量并重新编译生成 CSS。

1. 导入新变量

我们得更新__main.less，导入新的_variables.less 文件。

(1) 在__main.less 中，找到导入文件 bootstrap/variables.less 的代码。这是导入的第一个文件，在第 12 行。

(2) 修改这行代码，导入新的_variables.less 文件。很简单，删掉路径中的 bootstrap/，然后给文件名加上一个下划线。

```
@import "_variables.less";
```

(3) 接下来编译生成 CSS。如果你之前还没编译过，那赶紧把这个项目添加到你的编译器。

> 你的编译器可能需要你刷新文件视图才能看到这个新的_variables.less 文件，并将它添加到项目。（反正 CodeKit 需要。）

(4) 选择__main.less 进行编译。（如果有最小化或压缩选项，可以选中。）

(5) 将输出路径设置为 css/main.css。（这个文件是 index.html 中链接的样式表文件。）

> 如果在你的编译器里不好删除编译后文件名前的下划线，可以修改 index.html，让它链接到文件名带下划线的样式表文件。

(6) 编译！然后在浏览器中刷新 index.html。

如果这一步成功，那么链接和类为`btn-primary`的按钮颜色都会发生变化，因为它们使

用的都是变量@brand-primary 的颜色。

2. 编辑导航条变量

下面，我们来编辑设定导航条高度、颜色和悬停效果的变量。

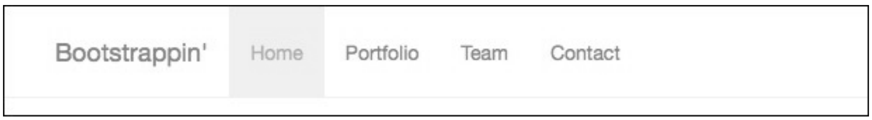
(1) 在_variables.less 中，搜索到下列变量，并修改为下列值。这样一来，就可以增加导航条的高度、把品牌色应用给链接，同时利用其他相关的颜色变量。

```
@navbar-height:                         64px;
@navbar-margin-bottom:                  0;
...
navbar-default-color:                   @gray;
@navbar-default-bg:                     #fff;
@@navbar-default-border:                @gray-light;
...
// Navbar links
@navbar-default-link-color:             @navbar-default-color;
@navbar-default-link-hover-color:       @link-hover-color;
@navbar-default-link-hover-bg:          @off-white;
@navbar-default-link-active-color:      @link-hover-color;
@navbar-default-link-active-bg:         @gray-lightest;
@navbar-default-link-disabled-color:    @gray-lighter;
@navbar-default-link-disabled-bg:       transparent;
```

(2) 保存修改、编译并刷新。

你应该看到下列有关导航条的新特性。

- 高度应该增加了 14px；
- 背景应该变成了白色；
- 底部边框稍微变暗了一些；
- 导航项的背景应该在悬停时稍暗一点；
- 导航项的背景应该在活动时稍暗一些；
- 链接文本的颜色应该在悬停和活动时变成品牌色，如下图所示：



接下来，我们把 Logo 放置到位。

2.8　添加 Logo 图片

在 img 文件夹里找到 logo.png 文件。你会发现这个图片非常大，有 900px 宽。在我们最终的设计中，这个 Logo 只有 120px 宽。因为多出来的像素将被压缩到较小的空间内，所

以这也是一种让图片在所有设备（包括视网膜屏设备）中保持清晰的一种简便方法。与此同时，这个图片的尺寸也针对 Web 进行了优化，只有 19 KB。

好，下面我们就把它放置到位并限制其宽度。

(1) 在编辑器中打开 index.html。
(2) 搜索到导航条标记中这一行代码：

```
<a class="navbar-brand" href="index.html">Bootstrappin'</a>
```

(3) 把这里的文本 Bootstrappin'替换成 `img` 标签，并添加 `alt` 和 `width` 属性。

```
<img src="img/logo.png" alt="Bootstrappin'" width="120">
```

> 一定要设置 `width` 属性，并将值设置为 120px。否则，图片会非常大。

(4) 保存 index.html 并刷新浏览器，你会看到 Logo 已经出现在了相应的位置上。

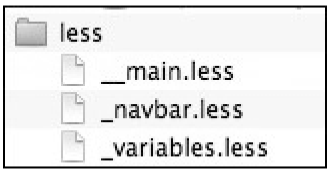


此时，导航条高度增加了，其底边也不再跟活动导航项的底边对齐了。这是因为 `navbar-brand` 元素周围设定了内边距，我们得再调整一下相应内边距的值。只需要以下简单几步。

(1) 在编辑器中打开 bootstrap/navbar.less 文件。
(2) 搜索到这个选择符及花括号：`.navbar-brand {`。
(3) 大约在第 150 行，可以看到如下代码：这里的 `padding` 就是我们要修改的。

```
.navbar-brand {
  float: left;
  padding: @navbar-padding-vertical @navbar-padding-horizontal;
```

(4) 因为我们想自定义这个文件，所以先将其另存为_navbar.less。
(5) 把它保存在 less 文件夹中，与__main.less、_variables.less 在一块。



(6) 下面把原来的声明注释掉，再添加我们自定义的内边距声明。在 LESS 中，注释只要在原先代码前添加两个斜杠即可。

```
// padding: @navbar-padding-vertical @navbar-padding-horizontal;
padding: 22px 30px 0 15px;
```

编译时，被注释掉的代码不会被编译成 CSS。

(7) 要看结果，先在_navbar.less 文件中作如上修改，然后保存。

(8) 然后，在__main.less 中找到并把导入原先 Bootstrap 的 navbar.less 文件的代码注释掉。然后，紧接着添加导入_navbar.less 文件的代码。

```
// @import "bootstrap/navbar.less";
@import "_navbar.less";
```

(9) 保存文件。确保把_navbar.less 添加到编译器，然后把__main.less 编译为 css/main.css。刷新浏览器，应该看到导航栏底边已经与活动链接的底边对齐了。（你还会发现 Logo 与 Home 链接之间的间隔也变大了。）

(10) 现在再用编辑器打开 main.css，搜索到`.navbar-brand{`。

如果 CSS 输出已经压缩，看到的结果应该是这样的：`.navbar-brand{float:left; padding:22px 30px 0 15px;`。

> 注释掉的那行声明并没有出现，因为 LESS 就没有把那行代码编译到 CSS 中！

LESS 的威力再次给我们留下了深刻印象。简单总结一下我们都干了什么：

- 我们并没有动原始的 bootstrap/navbar.less 文件，因此如果需要可以随时恢复原状；
- 我们完全用自己的定制版替换了这个文件，只是__main.less 里面的注释为我们留下了改动的痕迹；
- 我们还在_navbar.less 中注释掉一行代码，这样就可以知道修改了什么规则；
- 不过，由于我们使用的是 JavaScript 风格的单行注释，所有被注释掉的规则都不会被编译进最终的 CSS 文件。

换句话说，我们为自己留下了足够多的退路和相应的注释，同时并没有造成代码膨胀。不错！

2.9　调整导航项内边距

现在，我们来调整一下导航项，以便链接文本与 Logo 位于同一基线之上。

在_navbar.less 中，找到选择符`.navbar-nav`，这是导航项的父元素`ul`。在相应的规则里，可以看到嵌套的媒体查询。（关于嵌套媒体查询，可以参考 LESS 文档：http://lesscss.org。）

相关的行如下所示：

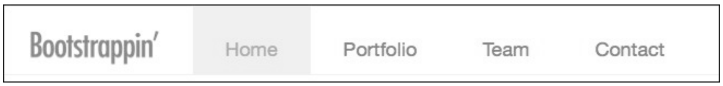
```
// Uncollapse the nav
  @media (min-width: @grid-float-breakpoint) {
    float: left;
    margin: 0;
    > li {
      float: left;
      > a {
        padding-top: ((@navbar-height - @line-height-computed) / 2);
        padding-bottom: ((@navbar-height - @line-height-computed) / 2);
      }
    }
  }
```

这里的变量@grid-float-breakpoint 指定了一个临界宽度，大于这个宽度，导航条就会扩展到与屏幕同宽；小于这个宽度，导航条就会折叠起来变成移动应用风格的响应式导航。（这个变量是在_variables.less 中定义的。）

现在，padding-top 和 padding-bottom 值都是动态计算的，以确保导航项中的文本垂直居中。而我们想增加上内边距，减少下内边距。与此同时，我们还要扩大导航项的间隔，再把字号增大一些。同样，把原来的代码使用单行注释注释掉，把需要的规则写在下面：

```
> a {
  // padding-top: ((@navbar-height - @line-height-computed) / 2);
  // padding-bottom: ((@navbar-height - @line-height-computed) / 2);
  padding: 30px 30px 14px;
  font-size: 18px;
```

保存、编译并刷新浏览器，可以看到如下效果：



够强吧？

再看图标的。

2.10 添加图标

现在轮到为导航项添加图标了。我们先直接使用 Boostrap 自带的 Glyphicons，然后再尝试一下 Font Awesome 这个大型图标库。

> 关于 Glyphicons，建议大家先看看 Bootstrap 文档：http://getbootstrap.com/components/#glyphicons。

稍后你会看到可用的图标，以及在 HTML 中使用 span 标记及 Glyphicon 类的约定。先从 Home 导航项开始。

(1) 要给 Home 导航项添加 Glyphicon Home 图标，只要在文本之前添加一个 span 标签，再加上特定的类即可：

```
<li class="active">
  <a href="index.html">
    <span class="glyphicon glyphicon-home"></span> Home
  </a>
</li>
```

> 注意这里在 span 标签和 Home 之间人为加了一个空格。

(2) 保存并刷新浏览器。一切顺利的话，应该能看到图标！



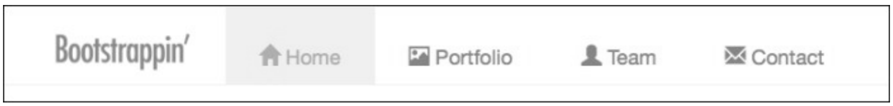
(3) 如果你没看到图标，请确保：

- Glyphicon 字体位于 fonts 文件夹；
- 变量文件_variables.less 中的@icon-font-path 值正确，我的是@icon-font-path: "../fonts/";。

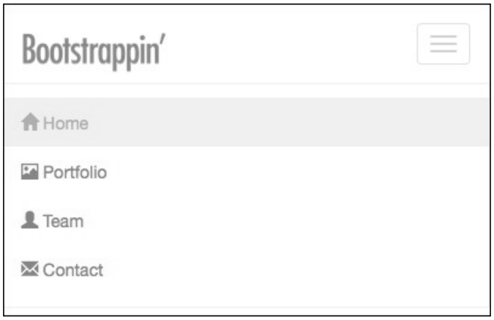
(4) 假设一切顺利，我们接着给剩下的导航项添加图标。以下依次是 Portfolio、Team 和 Contact 的图标对应的标签：

```
<span class="glyphicon glyphicon-picture"></span>
<span class="glyphicon glyphicon-user"></span>
<span class="glyphicon glyphicon-envelope"></span>
```

(5) 保存并刷新。应该看到下面展示的效果：

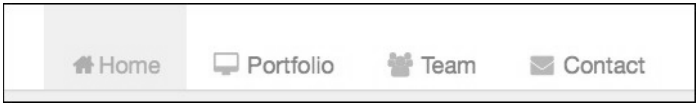


(6) 以及下面折叠后的响应式效果：



还不错。

当然，这里的图标与设计方案中的并不完全一致。设计方案中的是这样的：



Bootstrap 中免费使用的 Glyphicons 不包含计算机显示器和群组图标。此外，我们也没有从 Glyphicons 图标中找到适合页脚中社交媒体链接的图标。

好在，我们还有别的选择。下面就来看一看。

2.11　添加 Font Awesome 图标

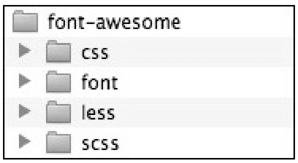
在本书写作时，Font Awesome 中包含 361 个图标，是 Glyphicons 的 Bootstrap 版本的近两倍。Font Awesome 图标是免费、开源的，而且也很方便在 Bootstrap 中使用。Font Awesome 的主页位于：

http://fortawesome.github.io/Font-Awesome/

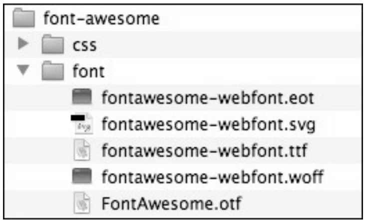
下面我们就来使用 Font Awesome 图标。

(1) 打开 Font Awesome 主页，即 http://fortawesome.github.io/Font-Awesome/，单击大大的 Download 按钮。

(2) 解压缩下载到的文件，可以看到以下文件结构：

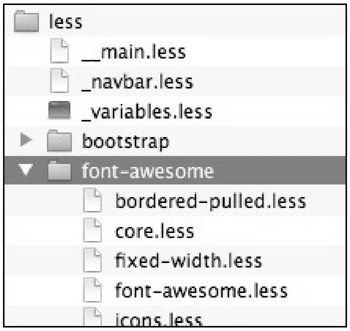


(3) 在 font 文件夹中，可以看到 Font Awesome 图标字体文件。



(4) 把所有字体文件复制粘贴到项目的 fonts 文件夹中，与 Glyphicons 字体文件在一块。

(5) 下面，我们要复制 Font Awesome 的 less 文件到我们项目的 less 文件夹。在项目的 less 文件夹中创建一个子文件夹，命名为 font-awesome，把 Font Awesome 的 less 文件复制过来。



(6) 接下来，要在__main.less 文件中导入 font-awesome.less 文件，以便将字体规则编译到样式表中。我们就在导入 Glyphicon 字体的下面添加 `import` 语句：

```
@import "bootstrap/glyphicons.less";
@import "font-awesome/font-awesome.less";
```

(7) Font Awesome 的 less 文件中包含一个变量，指定了 Font Awesome 字体的路径。我们得确保该路径与项目文件夹结构一致。打开 Font Awesome 变量文件 font-awesome/variables.less，确保`@fa-font-path` 变量的值是`../fonts`；如下所示：

```
@fa-font-path:    "../fonts";
```

> 这里的路径是相对于编译得到的 CSS 文件而言的，它位于 css 文件夹中。

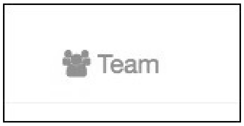
(8) 保存并编译 CSS。

(9) 现在，在 index.html 中，更新 Team 导航项的图标，使用 Font Awesome 名为 `fa-group` 的图标，此外还要加一个独立的 `fa` 类，最后再加上我们自己通用的 `icon` 类：

```
<span class="icon fa fa-group"></span> Team
```

(10) 保存 index.html，然后刷新浏览器。

如果一切顺利，应该看如下结果：



> 如果你看到的是一个奇怪的图标符号，或者什么也没看到，那说明 Web 字体并没有应用。这时候，要检查一下图标类是否正确（包括 fa 类），确保 Font Awesome 字体都在 fonts 文件夹中，而且 font-awesome/variables.less 中的路径也没有问题。

祝贺你——你有了双倍的图标！

此时此刻，我们可以同时使用 Glyphicons，也可以完全弃用它。为了减少代码冗余，咱们还是弃用它，只使用 Font Awesome 字体吧。为此只需要两步：

(1) 把__main.less 中导入 Glyphicons 字体的代码注释掉。

```
// @import "bootstrap/glyphicons.less";
@import "font-awesome/font-awesome.less";
```

(2) 修改 index.html 中图标标记的类，改为 Font Awesome 图标对应的类名。

查看 Font Awesome 图标的页面 http://fortawesome.github.io/Font-Awesome/icons/，可以看到图标对应的类名。根据设计方案，我们需要以下图标：

```
<span class="icon fa fa-home"></span> Home
<span class="icon fa fa-desktop"></span> Portfolio
<span class="icon fa fa-group"></span> Team
<span class="icon fa fa-envelope"></span> Contact
```

结果如下：



2.12 调整导航项图标颜色

发现没有？图标看起来比旁边文本显得粗大一些。虽然颜色一样，但图标看起来似乎更抢眼。下面我们就来调整图标的颜色，让它更浅一点。

(1) 在编辑器中打开_navbar.less。

(2) 搜索到选择符.navbar-default，我们是通过这个类为导航条应用默认样式的。这个选择符应该在// Alternate navbars 注释的下面。

(3) 在嵌套的规则中，再找到选择符.navbar-nav 和> li >，我们就在这里调整图标的颜色。

(4) 在定义导航项默认链接颜色的声明下面，我们要嵌套一个针对图标的规则，把图标颜色设置得更浅一些，因此要用到变量@gray-light：

```
.navbar-nav {
  > li > a {
    color: @navbar-default-link-color;

    .icon { // added rule set
      color: @gray-light;
    }
```

> 通用的 icon 类为我们选择图标提供了方便。

> 这里添加的的单行注释// added rule set，可以让我们将来方便地找到自己添加的规则。

(5) 此外，我们还得设定这些图标仍旧共享相同的悬停和活动颜色：@brand-primary。为此还要在我们刚刚添加的规则下方，再添加 icon 类。就在伪选择符&:hover 和&:focus 后面，再添加两个针对图标的组合选择符：

```
&:hover,
&:focus,
&:hover .icon, // added selector
&:focus .icon { // added selector
  color: @navbar-default-link-hover-color;
  background-color: @navbar-default-link-hover-bg;
}
```

再像下面的代码片段一样，针对活动链接添加针对图标的选择符：

```
> .active > a {
  &,
  &:hover,
  &:focus,
  .icon, // added selector
  &:hover .icon, // added selector
  &:focus .icon { // added selector
    color: @navbar-default-link-active-color;
    background-color: @navbar-default-link-active-bg;
  }
}
```

(6) 添加完这些选择符之后，保存文件，编译 CSS，刷新浏览器。你应该看到图标的颜色比默认的灰色浅了一些，但在悬停和活动状态下，仍然与链接文本的颜色相同。



导航条就这样设计完成了——或者说，基本完成了。因为我们在不经意间又制造了一个不得不解决的小麻烦。在继续之前，必须先把它及时处理掉。

2.13 调整响应式导航条断点

导航条在添加了 Logo 图片，增大了导航项，添加了图标之后，宽度也相应增加了。那么这也给我们的响应式设计造成了一个问题。试着从宽到窄（大约 480px）来回缩小窗口，就可以碰到导航项一下子被挤到 Logo 下方的情况。



怎么回事？因为在视口介于 768px 和 991px 之间时，导航对于它的容器来说太宽了。前面两个数值正是 Bootstrap 变量`@screen-sm-min` 和`@screen-md-min` 的值。

我们知道，变量`@grid-float-breakpoint` 决定了导航条在视口多宽的时候折叠起来。可以在_variables.less 文件的`// Grid system` 部分找到这个变量。

```
// Point at which the navbar stops collapsing
@grid-float-breakpoint:     @screen-sm-min;
```

我们需要调整这个断点，以便在视口宽度达到下一个更宽的断点（`@screen-md-min`）前，让导航条仍旧保持折叠。更新断点变量后的结果如下：

```
@grid-float-breakpoint:     @screen-md-min; // edited
```

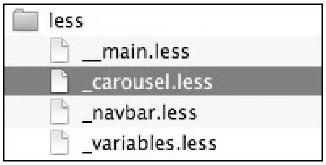
保存，编译并刷新。你会发现这一次导航条在到达下一个断点之间，始终保持折叠状态。

问题解决啦！接下来该调整传送带了。

2.14 调整传送带

关于传送带，我们主要还是使用 Bootstrap 默认的样式，同时对几个比较重要的地方进行定制。为此，我们复制一份 Bootstrap 的 carousel.less 文件，然后再进行修改。

(1) 复制 bootstrap/carousel.less，将其保存到 less 文件夹并命名为_carousel.less。



(2) 更新__main.less，用导入这个文件的代码代替导入 Bootstrap 中相应文件的代码：

```
@import "_carousel.less";
```

(3) 把_carousel.less 一开头的注释改成如下所示：

```
//
// Customized Carousel
// --------------------------------------------------
```

下面就可以开始定制了！

2.14.1　把控件改成使用Font Awesome图标

如果你在上一节就把 Glyphicons 字体删除了，那就会发现传送带左右两侧的“上一个”和“下一个”控件不见了。因为这两个图标就来自 Glyphicons。我们可以让 Font Awesome 图标取而代之。

(1) 首先，更新 index.html 中的图标标记。找到带有 `carousel-control` 和 `left`、`right` 类的链接：

```
<a class="left carousel-control" ...
```

(2) 将 `span` 标签中的类修改为通用的 `icon` 类，再加上 Font Awesome 图标类，如下所示：

```
<span class="icon fa fa-chevron-left"></span>
...
<span class="icon fa fa-chevron-right"></span>
```

(3) 接下来，在_carousel.less 中添加新类选择符。先找到`// Left/right controls for nav` 注释下的`.carousel-control` 选择符，在其嵌套规则中找到注释`// Toggles`，然后添加我们的`.icon` 类和注释：

```
// Toggles
  .icon-prev,
  .icon-next,
  .glyphicon-chevron-left,
  .glyphicon-chevron-right,
  .icon {  // added
    position: absolute;
    top: 50%;
    z-index: 5;
```

```
      display: inline-block;
    }
      .icon-prev,
      .glyphicon-chevron-left,
      &.left .icon { // added
        left: 20%; // edited was 50%
  }
      .icon-next,
      .glyphicon-chevron-right,
      &.right .icon { // added
        right: 20%; // edited was 50%
  }
```


关于以上修改的说明如下：

- 通过添加基本的 icon 类，我们可以使用任意图标，样式都不会出问题；
- &.left 和&.right 的写法是在嵌套层次中回溯一层，编译之后分别是.carousel-control.left 和.carousel-control.right；
- 修改 left:和 right:定位的值之后，让图标更靠近传送带两侧的边界。

保存，编译，刷新。新的 Font Awesome 图标应该粉墨登场了。



下面，我们对传送带进行一番修饰。

2.14.2　添加上、下内边距

先给.carousel 元素添加一点上、下内边距，并将背景色设置为@gray-lighter。为了将来方便查找，我在新增或编辑过的行尾都会加上注释。（当然，这些注释不会编译到最终的 CSS 里。）

```
.carousel {
  position: relative;
  padding-top: 4px; // added
  padding-bottom: 28px; // added
  background-color: @gray-lighter; // added
}
```

保存并编译后，透过新添加的内边距，可以看到传送带中图片上、下方的亮灰色背景。这样就似乎有了一个框，将图片与其上、下的元素隔离开来。此外，我们还要利用多出来的下内边距重新定位传送带指示器，让它更显眼一些。

不过，我们得先让图片在所有情况下都能自动拉伸填充所有空间。

2.14.3　强制图片全宽

无论屏幕多宽，我们都想让图片始终保持与屏幕同宽。因为传送带中的图片宽度为 1600px，基本可以涵盖大多数屏幕。如果屏幕宽度超过 1600px，那右侧就会出现间隙。



强制让图片在 1600px 以上的屏幕中也保持全宽，可能导致轻微的像素失真；但如果图片足够大，变形也不致于太明显。

> 如果时间允许，我们可以采用响应式图片方案，即在小屏幕上加载小图片，给大屏幕使用大图片。附录 B 给出了响应式图片的解决方案，大家可以参考。

现在，我们只要在文件中新增两行代码即可。.carousel 选择符下方，就是我们要添加声明的嵌套规则。在.carousel-inner 选择符下面，是针对传送带图片的规则，有一个混入用于确保图片响应，能够适应小屏幕。在此，我们也可以强制图片适应大屏幕，即设定一个 min-width::

```
.carousel-inner {
  ...
  > .item {
    ...
    > img,
    > a > img {
      .img-responsive();
      line-height: 1;
      min-width: 100%; // added
    }
  }
  ...
}
```

作出这个调整后，无论把浏览器窗口拉得多宽，图片都会跟着变宽。

接下来，需要限定传送带的最大高度。

2.14.4 约束传送带的高度

我们发现，在中大型屏幕中，传送带变得太高了。根据设计方案，传送带的高度应该大致在 440px。简单，我们只要在图片的父元素.carousel-inner > .item 上添加这个限制即可：

```
.carousel-inner {
  ...
  > .item {
    ...
    max-height: 640px; // added
```

因为.carousel-inner 元素有一条规则是 overflow: hidden，而.item 元素又被限定了高度，所以在图片高度超过最高限制后，其下面的部分会被隐藏起来。

在这一步的基础上，我们可以继续使用嵌套媒体查询（LESS 的另一个方便特性），再利用 Bootstrap 的中、大断点变量，分别调整一下屏幕宽度过宽时图片的垂直定位，从而保证我们的作品处于焦点位置。为此要用到以下代码：

```
> img,
> a > img {
  ...
  @media (min-width: @screen-md-min) {
    margin-top: -40px;
  }
  @media (min-width: @screen-lg-min) {
```

```
    margin-top: -60px;
  }
}
```

保存、编译并刷新。你会发现这时候的传送带已经基本成形，无论宽屏还是窄屏，都有模有样了。

屏幕窄时，是这样的：



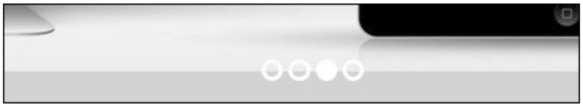
屏幕宽时，是这样的：



下面，该调整传送带指示器了。

2.14.5　重定位指示器

传送带指示器的作用是向用户显示一共有几张幻灯片，当前幻灯片是第几张。现在，指示器很难被看清楚，仔细看才会发现它位于作品图片底部中间。



下面我们把指示器放到它应该在的位置上：图片下方。

(1) 在_carousel.less 中，搜索到选择符`.carousel-indicators`，位于文件大约三分之二的位置，该位置有如下注释：

```
// Optional indicator pips
```

看看这个元素垂直方向的定位距离：

```
.carousel-indicators {
  position: absolute;
  bottom: 10px;
```

(2) 我们打算把它们挪到几乎靠近底边的位置，进入前面添加内边距制造出来的阴影区。调整底部定位的值可以做到。此外，还需要同时把下外边距重置为 0。

```
.carousel-indicators {
  position: absolute;
  bottom: 0; // edited
  margin-bottom: 0; // added
```

(3) 保存、编译，然后试一试。或许你也发现了，在小屏幕上，指示器的位置是我们想要的。而在大屏幕上，指示器仍然还在它原来的位置上。为什么？原来在适用的某个媒体查询里还有一条规则管着它呢，这个媒体查询就在文件最末尾。

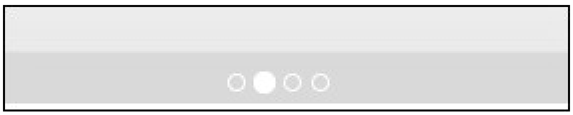
(4) 找到文件末尾的代码，在针对平板及以上屏幕的代码处：

```
// Move up the indicators
  .carousel-indicators {
    bottom: 20px;
  }
```

这是把指示器向上调的，现在我们用不着了。只要把它注释掉，就不会被编译到 CSS 中了。

```
// .carousel-indicators {
//   bottom: 20px;
// }
```

这样就可以达到我们目的。现在，指示器在各种屏幕上的位置都始终如一了。



接下来，我们调整指示器的外观，让它们更大，更显眼一些。

2.14.6　调整指示器外观

我们要使用灰色相关的变量，把传送带指示器调整得更显眼一些。除了调整颜色，也要把它们弄大一点。就从_variables.less 文件开始吧。

(1) 在_variables.less 中，位于`@carousel-control` 相关变量后面，可以看到两个以`@carousel-indicator` 开头的变量：这两个颜色用于默认状态下指示器的边框，还有活动状态下指示器的填充。

```
@carousel-indicator-active-bg:          #fff;
@carousel-indicator-border-color:       #fff;
```

(2) 我们在这里添加一个默认的背景颜色，并使用`@gray-light` 作为默认状态下指示器的填充色：

```
@carousel-indicator-bg:           @gray-light;
```

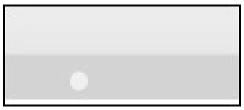
(3) 然后，修改活动状态下的背景色：

```
@carousel-indicator-active-bg:    @gray-lightest;
```

(4) 最后，把 `border-color` 设置为透明：

```
@carousel-indicator-border-color: transparent;
```

(5) 保存、编译并刷新。



至此，除了让活动状态下的指示器不可见，其他样式都就绪了。

下面，再打开_carousel.less。

(1) 在_carousel.less 中，找到`.carousel-indicators` 下面的第一组规则：

```
.carousel-indicators {
   position: absolute;
   ...
```

(2) 找到嵌套其中的 `li` 选择符。在这里需要修改几个值：

- 把 `width` 和 `height` 增大到 16px；
- 删除外边距；
- 添加 `background-color` 声明，值设置为新变量`@carousel-indicator-bg`；
- 删除边框线（前面把边框变量设置为透明，就是为了这里安全）；
- 为所有修改和新增添加注释，如下所示。

```
li {
  display: inline-block;
  width:  18px; // edited
  height: 18px; // edited
  // margin: 1px; // edited
  text-indent: -999px;
  background-color: @carousel-indicator-bg; // added
  // border: 1px solid @carousel-indicator-border-color;
  border-radius: 10px;
  ...
```

(3) 注意下面这两个针对 IE8-9 的手法，它们在这两个浏览器中为指示器提供背景颜色。因为我们已经给所有指示器都提供了背景色，所以这几行代码就没有必要了。把它们都注释掉吧，否则它们会干扰前面代码段中声明的背景色。

```
// background-color: #000 \9; // IE8
// background-color: rgba(0,0,0,0); // IE9
```

(4) 接下来，再注释掉.active 选择符下面的 margin、width 和 height，因为我们不希望活动状态下的指示器变大（当然也不希望它们缩小到 12px）。

```
.active {
  // margin: 0; // edited
  // width:  12px; // edited
  // height: 12px; // edited
  background-color: @carousel-indicator-active-bg;
}
```

(5) 最后，与.active 选择符并列添加一个:hover 选择符，增加悬停效果：

```
li:hover, // added
  .active { ...
```

(6) 保存，编译并刷新。看看结果吧！



传送带的调整工作做完啦！一路下来，我们也学到不少东西：很多 Bootstrap 的约定，还有一些 LESS 的。

从下一节开始，事情就越来越简单了。

2.15 调整分栏及其内容

下面我们来调整一下位于标题 Welcome!、Recent Updates 和 Our Team 下面的三个内容块。

首先，为每个块中的按钮添加圆圈箭头图标。还记得我们可以使用 Font Awesome 吧。

(1) 查看 Font Awesome 文档：http://fortawesome.github.io/Font-Awesome/icons/；

(2) 可以看到我们想使用的图标：

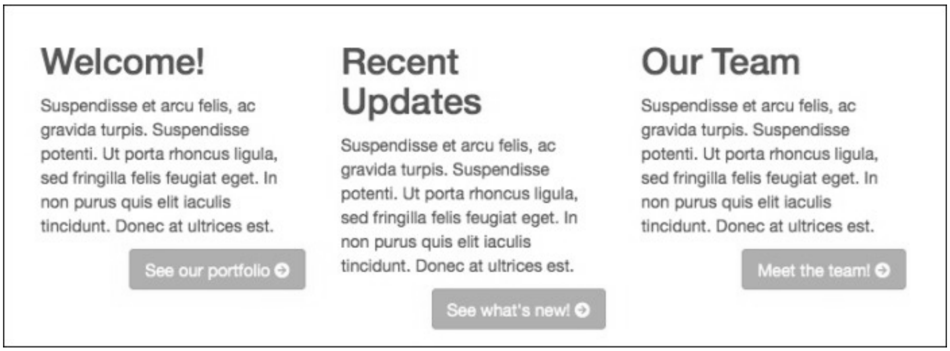


(3) 在 index.html 中，为每个链接添加带有适当类的 span 标签。下面是为第一个链接添加代码后的结果，为清晰起见，加了回车换行。

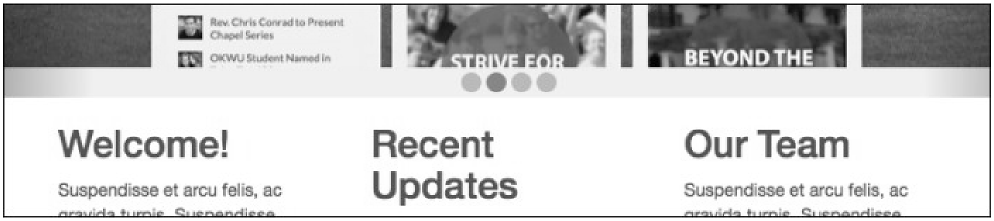
```
<p>
  <a class="btn btn-primary pull-right" href="#">
    See our portfolio <span class="icon fa fa-arrow-circle-right"></span>
  </a>
</p>
```

(4) 每个链接都如此。

这样，每个按钮上就都有了相同的图标了。



再给文本块与传送带之间增加一些垂直内边距，现在太紧了。

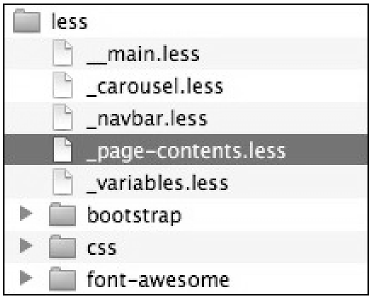


当前的问题是，相关的样式放在哪里最合适？为页面中的内容添加额外的内边距在目前和以后都是经常可能发生的，所以我们最好创建一个 LESS 文件，用以保存这些及其他改动。（巧的是，我们也正需要一个这样的文件，用以添加额外的、更重要的响应式调整，所以看来真有必要创建一个新文件了。）

2

(1) 创建一个新文件，命名为_page-contents.less。

(2) 把它保存到 less 文件夹中，与其他自定义 LESS 文件放在一起。



(3) 在文件中添加以下注释：

```
//
// Page Contents
// --------------------------
```

(4) 然后，写一个让人一目了然的类名，再加上适当的内边距——包括下内边距：

```
.page-contents {
  padding-top: 20px;
  padding-bottom: 40px;
}
```

(5) 保存文件。

(6) 把_page-contents.less 导入到__main.less 中。在这里，我把导入代码放到文件最下面`// Utility classes` 注释的上面，同时也加上注释：

```
// Other custom files
@import "_page-contents.less";
```

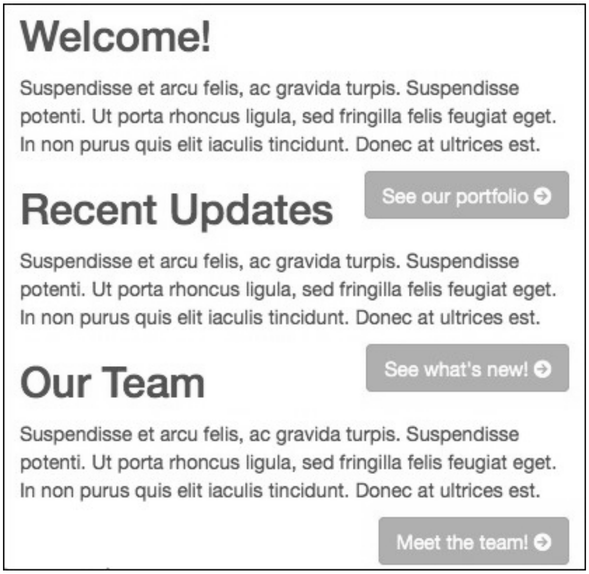
(7) 保存并编译。

(8) 下面再在标记中添加必要的类。打开 index.html，给带有 `container` 类的 `div` 元素添加 `page-contents` 类，就在 `homepage-feature` 传送带的结束标签处：

```
</div><!-- /#homepage-feature.carousel -->
<div class="page-contents container">
<div class="row">
```

(9) 保存并刷新浏览器，应该看到内边距已经加上了。

接下来，我们再对窄屏幕下这些块的效果进行调整。如下图所示，在一栏布局时，标题并没有清除相邻的按钮。



这个问题有点棘手。我们本想给每个包含块 `div` 添加一个 `clearfix` 类，但不行，因为在视口在 768px 及以上宽度时，需要让这些块都浮动起来，谁也不能清除谁。

这时候就要用到媒体查询了。因为三栏布局是从`@screen-sm` 断点，也就是 768px 开始的。那我们可以用媒体查询来设置在视口比这个断点小 1 像素时，就给文本块应用清除规则。而比`@screen-sm` 小 1 像素的正是特殊断点`@screen-xs-max`。在_variables.less 中其他`@screen` 变量下面，可以看到它：

```
// So media queries don't overlap when required, provide a maximum
@screen-xs-max:              (@screen-sm - 1);
@screen-sm-max:              (@screen-md - 1);
@screen-md-max:              (@screen-lg - 1);
```

这里的`@screen-xs-max` 断点正是我们需要的，因为它的值恰好比`@screen-sm` 断点小 1 像素。

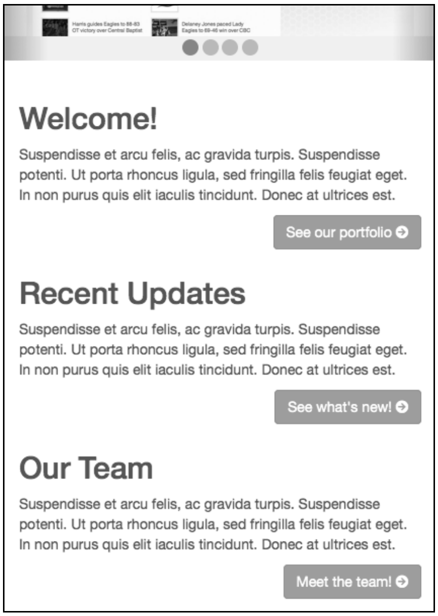
> 不能在这里使用@screen-sm-min 变量，因为多出这 1 像素，会导致三栏分别都是近似三分之一容器的宽度，而三栏加到一起又太宽，不能并排浮动，只能垂直堆叠。经过测试，多出这 1 像素会在视口宽度为 768px 时导致布局失控。因此，一定要使用@screen-xs-max 变量！

除此之外，最好再给这几栏添加一些下内边距，让它们垂直堆叠时，相互之间有点空隙。

在我们的媒体查询中，将使用 CSS 2 的属性选择符来选择类中包含 col-的所有元素，从而让同一组规则能够应用给任何尺寸的分栏：


```
.page-contents {
...
@media (max-width: @screen-xs-max) {
  [class*="col-"] {
    clear: both;
    padding-bottom: 40px;
  }
}
}
```

保存、编译并刷新。结果大大不同！



效果好看多了！

下面该来修饰页脚了。

2.16　修饰页脚

页脚最主要的功能就是罗列社交图标。就用 Font Awesome！

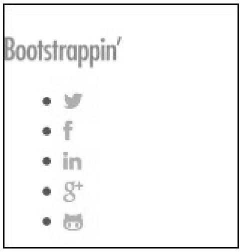
查询 Font Awesome 文档，可以在 Brand Icons 部分找到我们想到的图标：

http://fortawesome.github.io/Font-Awesome/icons/#brand

那么，只要把页脚中的社交链接替换成带有相应类的 span 元素即可。

```
<ul class="social">
  <li><a href="#" title="Twitter Profile"><span class="icon fa
    fa-twitter"></span></a></li>
  <li><a href="#" title="Facebook Page"><span class="icon fa
    fa-facebook"></span></a></li>
  <li><a href="#" title="LinkedIn Profile"><span class="icon fa
    fa-linkedin"></span></a></li>
  <li><a href="#" title="Google+ Profile"><span class="icon fa
    fa-google-plus"></span></a></li>
  <li><a href="#" title="GitHub Profile"><span class="icon fa
    fa-github-alt"></span></a></li>
</ul>
```

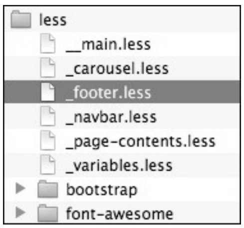
替换之后的标记让原来的社交链接变成了图标链接：



为了让这些图标水平排列并居中，执行下列操作。

(1) 创建一个新文件_footer.less 来保存相关样式。

(2) 把这个文件保存到 less 文件夹中：



(3) 在__main.less 中添加导入这个文件的代码：

```
    // Other custom files
    @import "_page-contents.less";
    @import "_footer.less";
```

接下来就可以写样式了。我们先把样式摆出来，然后再解释。

调整页脚的样式如下：

```
ul.social {
  margin: 0;
  padding: 0;
  width: 100%;
  text-align: center;
  > li {
    display: inline-block;
    > a {
      display: inline-block;
      font-size: 18px;
      line-height: 30px;
      .square(30px); // see bootstrap/mixins.less
      border-radius: 36px;
      background-color: @gray-light;
      color: #fff;
      margin: 0 3px 3px 0;
        &:hover {
        text-decoration: none;
        background-color: @link-hover-color;
        }
    }
  }
}
```

下面来逐行解释。

- 去掉 `ul` 中默认的内、外边距；
- 将容器宽度拉伸到百分之百；
- 内容居中；
- 列表项显示为行内块，因此可以像文本一样居中；
- 链接也显示为行内块，从而可以填满有效空间；
- 增大字号和行高；
- 使用 Boostrap 的混入，将宽度和高度设计为 30px 见方；
- 要查看这个混入，打开 bootstrap/mixins.less，搜到 `.square`，可以看到下列代码：

```
// Sizing shortcuts
.size(@width; @height) {
  width: @width;
  height: @height;
}
.square(@size) {
  .size(@size; @size);
}
```

- ❑ border-radius 属性的值设置得足够大，以便图标及其背景呈现圆形；
- ❑ 设置背景色、前景色和外边距；
- ❑ 去掉悬停状态默认出来的下划线，同时把背景色改为浅灰色。

设置以上样式后，我们再给页脚添加一些上、下内边距，然后将内容居中，以便 Logo 居中显示在社交图标上面。

```
footer[role="contentinfo"] {
  padding-top: 24px;
  padding-bottom: 36px;
  text-align: center;
}
```

结果如下所示：



挺好——但愿跟你看到的一样！

2.17　接下来做什么

在实际做一个类似的项目之前，我强烈建议大家至少再做一件事。那就是花点时间优化你的图片、CSS 和 JavaScript。这些优化并不难。

- ❑ 压缩图片花不了多少时间，但却能解决导致图片臃肿的最大问题。本章中的图片都在 Photoshop 中使用了“保存为 Web 格式”，但或许你还是能够再把它们压缩一些。
- ❑ 此外，应该马上从__main.less 中删除那些不需要的 Bootstrap 的 LESS 文件，然后最小化 main.css。
- ❑ 最后，还要对 plugins.js 进行“瘦身”，把 Bootstrap 原来大而全的 bootstrap.min.js 替换成只包含我们用到的 carousel.js、collapse.js 和 transitions.js 的压缩版。然后再压缩最终的 plugins.js 文件。

做完以上三项优化，整个网站的体量大致将缩小一半。在速度就是生命的年代，既要考虑用户体验，又得考虑 SEO 排名，这么大幅度的优化可不得了。为了帮助大家完成优化，

附录 A 提供了针对这个项目的优化步骤。

此外，还有两个非常重要的可能性，或许你也应该考虑。

第一，我们可以通过实现响应式图片来进一步优化传送带图片。那些图片太大了，特别是对小屏幕而言，实在没有必要那么大。相反，如果是像视网膜屏这样的高密度屏，为保证显示效果清晰锐利，我们也应该提供高分辨率的图片。在附录 B 中，我们会实现 Scott Jehl 令人叫绝的 Picturefill 方案。

第二，我们知道，触摸屏设备的用户喜欢用手指来回扫屏切换传送带图片。在附录 C 中，我会向大家展示如何利用出色的 Hammer.js 插件，只几步就为传送带添加扫屏交互支持。

不过现在，我们可以先停一下，庆祝一番。

2.18　小结

下面我们来清点一下本章的知识要点。

- 我们以 HTML5 Boilerplate 可靠的 HTML 标记结构作为起点；
- 我们利用了 Bootstrap 的响应式导航条、传送带和网格系统；
- 我们自定义了一些 Bootstrap 的 LESS 文件；
- 我们还创建了自己的 LESS 文件，并将它们无缝集成到了项目中；
- 我们加入 Font Awesome，获得了双倍的图标；
- 我们实现了一个未来容易维护的网站，因为网站的文件组织非常稳健，注释非常完善——而且不会造成代码臃肿。

有了这些经验，你就真的可以让 Bootstrap 为你所用了：利用它可以迅速搭建网站框架，然后再对核心内容进行定制。在本书后面几章里，我们还将继续丰富你的经验。接下来，我们先考虑把本章的设计方案转换成一个 WordPress 主题。

3 WordPress 主题

这一章，我们来把第 2 章的个人作品站点变成一个 WordPress 主题。基于 Bootstrap 的主题其实很多。而我们一直关注把 Bootstrap 强大的 LESS 样式和 JavaScript 插件，以及 HTML5 Boilerplate 中的最佳实践结合在一起。因此选择一款符合上述要求的主题对我们是有利的。

Roots 主题就是理想的选择，它从一开始就将自己定位为 Bootstrap 驱动的、各方面都力求体现最佳实践（包括 HTML5 Boilerplate 及其他好的做法）的一个基础主题。因此本章就使用这个主题作为起点。

本章内容：

- 在 Roots 主题中集成我们自定义的 LESS 和 JavaScript 文件；
- 自定义主题模板文件，以提供我们主页传送带和分栏内容所需的标记；
- 利用强大先进的自定义栏目，为传送带项和主页分栏提供自定义栏目；
- 创建主页模板文件，将自定义栏目发布为期待的主页布局。

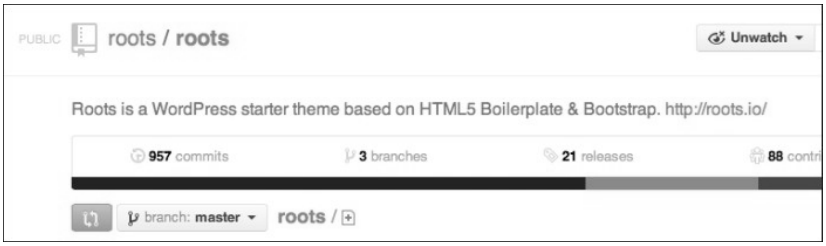
3.1 下载并重命名 Roots 主题

我们先下载 Roots 主题[①]。

(1) 打开 Roots 主题的主页：http://roots.io/。
建议大家花点时间在上面逛一逛，熟悉一下这个资源。（这是一个不错的生机勃勃的社区。）

(2) 点击 GitHub 的图标链接，进入该项目的 GitHub 页面。直接访问：https://github.com/roots/roots。

① 作为练习，大家可以直接使用源代码中提供的主题文件。如果使用下载到的 Roots 主题，则根据官方文档（https://roots.io/sage/docs/theme-installation/），在不使用 Bedrock 的情况下，需要在 wp-config.php 文件中添加一行代码：`define('WP_ENV', 'development');`。——译者注



(3) 下载 ZIP 文件。

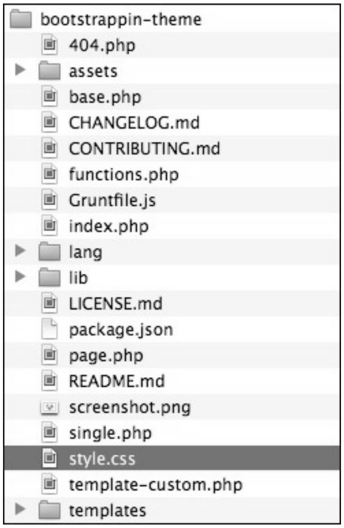
(4) 解压缩。



(5) 如下图所示，把解压得到的文件夹重命名为你想叫的主题名：



(6) 找到主文件夹中的 style.css 文件，然后在编辑器中打开它：



(7) 打开这个文件后，你会发现里面其实没有样式。站点的样式来自 assets 文件夹中的 css 文件夹，是由 Bootstrap 编译生成的。我们继续沿用 Bootstrap 的这种模式。而 style.css 在这里主要用于命名主题、列出贡献者名单、声明许可。

(8) 按照下面的样板修改主题名称：

```
/*
Theme Name:         Bootstrappin' Theme
Theme URI:          [your site URI]
Description:        A custom theme based on <a href="http://www.
                    roots.io">the Roots Theme</a>
Version:            1.0
Author:             [你的名字]
Author URI:         [你的 URL]

License:            [你的许可]
License URI:        [许可 URI]
*/
```

(9) 保存这个文件。

(10) 接下来添加一个自定义的屏幕截图，以便在 WordPress 的控制板中更容易找到它。

(11) 截取第 2 章示例站点的一张截图。（03_Code_BEGIN 的练习文件夹里有一张了。）

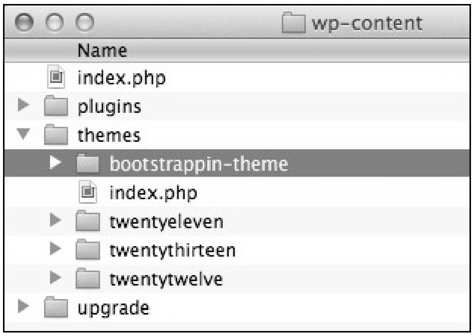
(12) 把 Roots 默认的屏幕截图替换成自定义的屏幕截图。

好了，现在可以安装 Roots 主题了。

3.2　安装主题

前面的修改暂时切断了与 Bootstrap 样式、脚本等的联系。接下来几步要恢复这些联系。为了让整个过程更真实有趣，我们可以先把主题安装上，运行起来。这样就可以随时测试了。

(1) 把新主题上传到 WordPress[1]的 themes 文件夹。（如果是在本地，复制过去就行了。）



(2) 在 WordPress 的控制板中，找到“外观 | 主题”，然后启用这个主题。如果你按照前面对这个主题进行了重命名，也换了屏幕截图，那么结果应该如下所示：

① 这里使用 WordPress 4.0.1 中文版截图。——译者注



(3) 点击“启用”，会打开启用主题页面，页面中会展示如下选项，相应的选择如下：

- **是否建立静态首页？**
  是
  我们要将它作为站点主页。
- **是否修改固定网址结构？**
  是
  这个选项是我们最开始都会设置的，这样页面的 URL 才会显示文章和页面的名称。

> 在我安装的 MAMP 中，我发现只有更新了固定链接结构，Roots 的主题路径才会生效。

- **是否修改上传文件夹？**
  否
  你可以选择将上传文件保存到另一个文件夹，也可以不改，使用默认文件夹。
- **是否建立导航菜单？**
  是
  这里设置的是顶部导航。
- **是否添加页面到菜单？**
  是
  当然啦，谢谢！



> 如果你看到提示说要确保.htaccess 文件可写，可以在 WordPress 站点的根目录（不是 themes 文件夹，而是包含 wp-content、wp-admint 文件夹的根目录）创建一个.htaccess 文件，并把权限设定为 664。

(4) 保存更改。

(5) 然后又会回到“外观 | 主题”管理页面。



(6) 单击“自定义”按钮。

(7) 然后，会打开一个可以设置该主题的页面。

- **站点标题和副标题**：修改副标题
- **导航菜单**：按需更改
- **静态首页**：按需更改

(8) 可以在右侧面板中看到默认的 Bootstrap 导航条、页面标题和一段文字。



恭喜，你的 Roots 主题安装就绪了。

> 如果看到导航条或文本没有应用默认的 Bootstrap 样式，则说明 Roots 主题的路径设置出了问题。一般来说，这时候只要在“设置 | 固定链接”中重新设置固定链接就可以了。

下面我们就从导航项开始。

3.3 配置导航条

本节，我们为站点页面添加导航条中的导航项，此外还会为图标添加相应的标记。

(1) 在 WordPress 的仪表板中，打开“外观 | 菜单”。



(2) 删除“示例页面”菜单项。

(3) 编辑“Home”菜单项，参照第 2 章中的 index.html 文件，在“导航标签”文本框中添加 Font Awesome 字体图标的 HTML 标记，完成后的内容如下：

```
<span class="icon fa fa-home"></span> Home
```



(4) 接下来再为其他页面创建导航项，目前只是创建简单的自定义链接，注意对应每个页面的标记要正确：



(5) 换句话说，要在“链接文字”文本框中添加的菜单项标记如下：

```
<span class="icon fa fa-desktop"></span> Portfolio
<span class="icon fa fa-group"></span> Team
<span class="icon fa fa-envelope"></span> Contact
```

(6) 添加完所有导航项之后，保存菜单。

> Font Awesome 图标现在还不可用。Bootstrap 和 Roots 主题默认并不提供这种字体。因此，到后面我们添加这种字体之后才能看到图标。

(7) 下面打开站点并刷新页面，就会看到添加到主导航的链接文本了。



下面该添加首页内容了。

3.4 添加首页内容

说到添加首页内容，两个方案会浮现在脑海中。

- **所见即所得地填充**：复制传送带和分栏的标记，粘贴到可视化编辑器里，然后上传图片，再按照标记结构把它们放置到位。
- **自定义栏目**：使用 WordPress 自定义栏目输入关键元素——传送带图片和分栏的内容，然后自定义模板文件，实现我们想要的标记结构。

第一个方案无法满足长远需要，因为所见即所得编辑器不适合处理复杂标记。但不管怎么样，这是目前来讲最快的方案，可以让我们快速上手。所以我们可以先易后难，到最后再引入自定义栏目的方案。

在本章的练习文件中，可以找到第 2 章的结果文件。我们要使用其中 index.html 文件的标记，使用其中的传送带的分栏结构。

(1) 打开第 2 章的 index.html，复制 `main` 标签中的所有代码，不包含 `main`。

- 首先是传送带：

  ```
  <div id="homepage-feature" class="carousel slide">
  ```

- 我们要选择所有标记，包括三个分栏和包含它们的类为 `row` 和 `container` 的 `div` 标签，这样就到了对应的`</main>` 标签那里。

(2) 复制上一步中选择的内容，但先不要粘贴到 WordPress 里。我们知道，WordPress 会自动为内容添加段落标记。为了避免触发这个行为，我们需要先清理一下复制的内容。

(3) 把复制的内容先粘贴到一个新打开的代码编辑器窗口中。

(4) 删除所有缩进，因为缩进会在我们把代码粘贴到 WordPress 后，给我们带来麻烦。选中整个代码块，让代码左移，直到顶到头为止。

(5) 然后删除独占一行的注释，主要是以下两行：

```
<!-- Wrapper for slides -->
<!-- Controls -->
```

(6) 删除所有空行，前面两行注释上方都有空行。

(7) 复制处理后的所有代码。

(8) 在 WordPress 仪表板中，单击“页面”，编辑名为 Home 的页面。

(9) 在粘贴之前，把可视化编辑器切换至文本编辑器（单击右上角的“文本”标签），如下图所示：



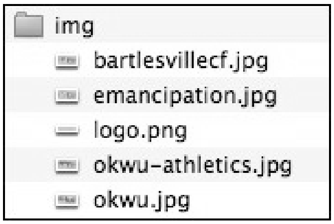
(10)　把标记粘贴到编辑框内。

这仅仅是开始，接下来我们要上传图片。

添加图片

下面上传图片，并将它们放到标记中合适的位置。

(1) 在第 2 章项目文件的 img 文件夹中，可以找到作品图片和 Logo，该目录截图如下：



(2) 在 WordPress 的 Home 页面编辑器中，单击“添加媒体”按钮上传图片。

(3) 选择并上传 4 张作品图片，如下图所示：



(4) 如下图所示，对每张图片进行设置。

- ❑ 填写相应的“替代文本”；
- ❑ 在“对齐方式”中选择“无”；
- ❑ 在“链接到”中也选择“无”（当然，实际上每张作品图片都应该链接到一个页面）；
- ❑ 在“尺寸”中选择“完整尺寸”。



(5) 把完整大小的 4 张图片分别插入到标记中的相应位置，将原先的标记替换为新的。

比如下面这行原来的代码：

```
<img src="img/okwu.jpg" alt="OKWU.edu Homepage">
```

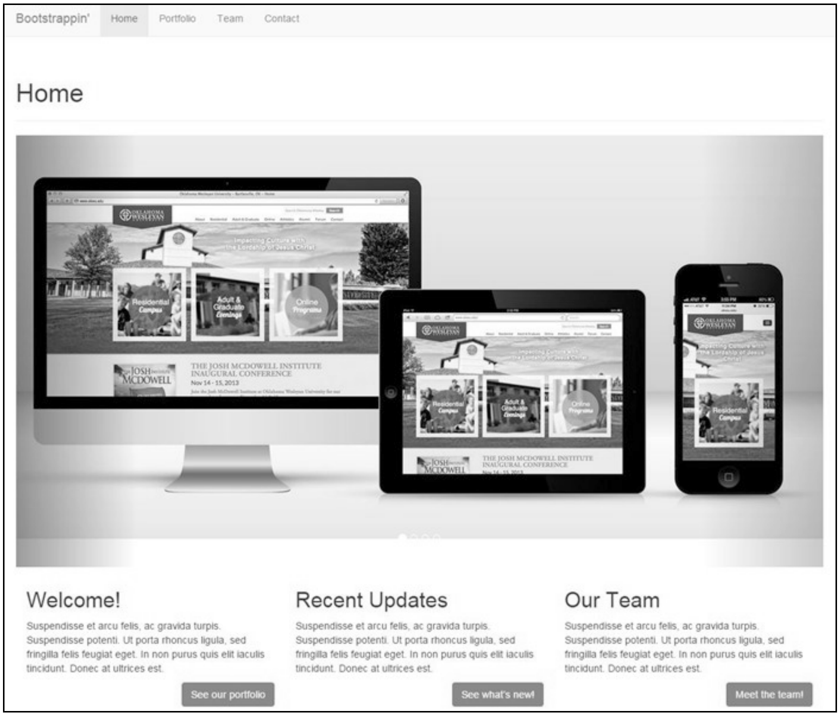
应该变成类似下面这样的代码（其中的 `src` 属性会因你的 WordPress 安装位置而有所不同）：

```
<img src="http://localhost/wordpress/wp-content/uploads/2014/12/okwu.jpg"
alt="OKWU.edu Homepage" width="1600" height="800" class="alignnone size-full
wp-image-18" />
```

每张图片都如法炮制。

(6) 上传并插入作品图片后，可以看一看效果了。

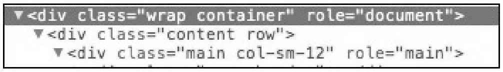
(7) 打开站点的 Home 页面，刷新一下。如果一切顺利，应该能看到如下面屏幕截图所示的结果：



主页内容已经就位了。不过 Boostrap 默认的颜色、传送带尺寸，以及其他细节都在。稍后我们会修复这些细节，在此之前，我们要先解决两个模板问题。

主页的标题不是我们想要的——这是标准 Roots 页面模板的配置。另外，传送带的宽度与

后面三栏的宽度一样。查看页面元素会发现，页面模板把整个页面内容封装在了一个使用 Boostrap 的 `container` 类的元素中，用以约束页面内容的宽度。



我们要调整模板，删除页面的标题，同时把传送带从封装它的元素中解放出来。

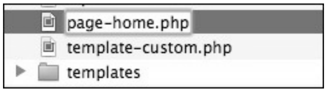
3.5 自定义页面模板

对一般的页面而言，我们一定想显示页面的标题。当然，主页比较特殊，我们可以给它写一个模板。

接下来，我们就定义一个页面模板。在这个模板中，删除原来的标题，突出页面内容。页面内容主要是传送带和三栏的结构化标记。

(1) 在 themes 文件夹中当前主题的目录下，找到 template-custom.php 文件。这个文件是一个模板的示例，使用它可以快速上手。

(2) 复制这个示例模板并将其重命名为 page-home.php，结果如下图所示：

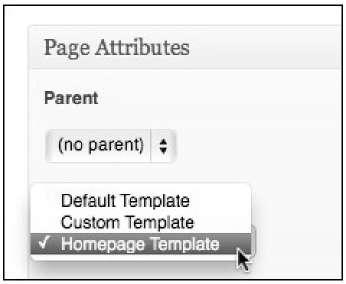


(3) 用编辑器打开新建的 page-home.php 文件，修改开始位置的注释，将模板名字改为 Homepage Template：

```
/*
Template Name: Homepage Template
*/
```

(4) 保存文件。

(5) 返回 WordPress 的仪表板，编辑 Home 页面。把它的页面模板修改为新创建的 Homepage Template。



(6) 更新页面。

(7) 不用刷新浏览器查看结果，因为不会有什么变化。还是接着修改模板吧。

(8) 在编辑器中切换到打开的 page-home.php 文件，找到下面这行代码：

```
<?php get_template_part('templates/page', 'header'); ?>
```

(9) 这是模板中仅有的两行 PHP 代码的第一行，其作用是插入页面标题。（这行代码导入的模板文件是 templates 文件夹中的 page-header.php。）想修改页面标题，只要把这行代码删除或者注释掉就行了。在此我们用单行注释把这行引入标题模板的代码注释掉：

```
<?php // get_template_part('templates/page', 'header'); ?>
<?php get_template_part('templates/content', 'page'); ?>
```

(10) 保存这个文件，在浏览器中刷新查看 Home 页面。结果应该像下图一样，标题已经不见了：



我们迈出了第一步！第二步就是把传送带从现有的容器元素中解放出来。

3.6　理解 Roots 的基准模板

Roots 主题做了一件很地道的事，它把基本的布局元素从单个模板文件中给抽了出来，放到了一个叫 base.php 的文件中。下面我们就来讲这个模板。

首先，我们注意到 base.php 引用了 lib 文件夹中的 config.php 文件定义的基本布局指令。我们打开这个 config.php 文件看一下，但不会改动它。关于这个文件的更多信息，可以参考下面两个页面。

- Roots 101：http://roots.io/roots-101
- An Introduction to the Roots Theme Wrapper：http://roots.io/an-introduction-to-the-roots-theme-wrapper/

当下，我们只要知道自己可以决定主栏和侧边栏使用什么类，以及哪些页面不包含标准的侧边栏就可以了。

在 config.php 中搜索“main class”，可以找到设置 main 和 sidebar 类的代码。如果页面包含侧边栏，则 main 元素会被赋予 col-sm-8 类，将其宽度限制为容器宽度的三分之二；侧边栏则会被赋予 col-sm-4 类。如果没有侧边栏，主栏的类就是 col-sm-12，与容器同宽。

```
/**
 * .main classes
 */
function roots_main_class() {
  if (roots_display_sidebar()) {
    // Classes on pages with the sidebar
    $class = 'col-sm-8';
  } else {
    // Classes on full width pages
    $class = 'col-sm-12';
  }
  return $class;
}
/**
 * .sidebar classes
 */
function roots_sidebar_class() {
  return 'col-sm-4';
}
```

以上代码中的类名表明，我们页面布局从单栏切换到多栏的断点是@screen-sm。想要修改断点，非常简单。同样地，想要修改主栏和侧边栏的宽度，也很简单，都是只修改类名就行。换句话说，对这一个文件的修改，可以影响整个站点中所有页面和文章的布局，而不必再组合使用多个模板文件了。乍一看不太好理解，但好处十分明显。

其次，config.php 中确定哪些页面显示侧边栏，哪些不显示的代码片段如下：

```
/**
 * Define which pages shouldn't have the sidebar
 * ...
 */
function roots_display_sidebar() {
  $sidebar_config = new Roots_Sidebar(

    array(
      'is_404',
      'is_front_page'
    ),
    ...
    array(
      'template-custom.php'
    )
  );
  return apply_filters('roots_display_sidebar', $sidebar_config->display);
}
```

默认情况下，头版页面（front page）、404 页面，以及其他使用自定义页面模板的页面都不会包含侧边栏。修改这两个数组，添加或删除页面或模板，就可以自定义站点的整体或个别页面的布局。

明白了这些，接下来用编辑器打开主题目录下的 base.php。

浏览一遍这个文件，会发现它做了以下几件事。

- 拉进了页面的`<head>`，见 templates 文件夹中的 head.php 文件。

```
<?php get_template_part('templates/head'); ?>
```

- 给出 `body` 标签和它的类。

```
<body <?php body_class(); ?>>
```

- 视情况拉进顶部的导航栏或带常规导航的头部，见 header-top-navbar.php 和 header.php 模板。

```
<?php
  do_action('get_header');
  // Use Bootstrap's navbar if enabled in config.php
  if (current_theme_supports('bootstrap-top-navbar')) {
    get_template_part('templates/header-top-navbar');
  } else {
    get_template_part('templates/header');
  }
?>
```

- 往底部看，它又拉进了脚部，见 templates 文件夹中的 footer.php 文件。

```
<?php get_template_part('templates/footer'); ?>
```

- 而在中间，则是定义页面内容结构的代码。
  - 先是一个容器和一行：

  ```
  <div class="wrap container" role="document">
    <div class="content row">
  ```

  - 然后是类为 `main` 的 `div`，通过 `roots_main_class` 方法决定列宽：

  ```
  <div class="main <?php echo roots_main_class(); ?>" role="main">
  ```

> 还记得吗？`roots_main_class` 方法是在 lib 文件夹下的 config.php 中定义的。

  - 使用适当模板展示当前页面内容：

  ```
  <?php include roots_template_path(); ?>
  ```

  - 关闭`.main` 的 `div` 标签：

```
</div><!-- /.main -->
```

■ 如果需要，则显示带有 Bootstrap 类的侧边栏：

```
<?php if (roots_display_sidebar()) : ?>
  <aside class="sidebar <?php echo
    roots_sidebar_class(); ?>" role="complementary">
    <?php include roots_sidebar_path(); ?>
  </aside><!-- /.sidebar -->
<?php endif; ?>
```

> 决定哪些页面应该带侧边栏以及侧边栏类名的，还是 config.php 文件。

■ 关闭两个 `div` 标签：`.conent` 和`.wrap`，它们分别还有 `row` 和 `container` 类。

```
  </div><!-- /.content -->
</div><!-- /.wrap -->
```

内容不少，概括如下：

- Roots 的 base.php 模板提供了导航条和页脚之间的所有内容，这些内容包含在一个带 Bootstrap `container` 类的元素中，以便约束其宽度，避免全宽；
- 我们需要让传送带全宽，并在合适的时候和地方再应用 `container` 类。

我们可以简单地从 base.php 中删除 `container` 类，然后相应地调整其他元素。但事实上只有首页才需要全宽，这样做反倒会弄乱标准模板，造成很多不必要的麻烦。

为此，可以换一种更有针对性的做法：针对首页新建一个模板。Roots 对这种做法支持得很好。

3.7 创建自定义的基本模板

Root 主题确实强大，虽然 base.php 文件为整个站点定义了基本布局，但我们仍然可以弄一个自己的基本模板，在必要的时候自定义基本布局的结构。下面我们就针对首页来做一次自定义。

(1) 复制 base.php 文件。

(2) 把副本命名为 base-page-home.php，因为前面已经设置页面使用 page-home.php 模板文件了，所以 Roots 会按同样的文件名来读取基本模板，如果找到这个文件，就将其作为首页的模板。



> 要了解基本模板选择函数的更多信息，参见 Roots 的相关文档：http://roots.io/an-introduction-to-the-roots-theme-wrapper/。

(3) 在编辑器中打开新的 base-page-home.php 文件。

(4) 找到带 wrap 类的 div 标签，去掉它的 container 类，结果如下：

```
<div class="wrap" role="document">
```

(5) 我们会在合适的时候使用 container 类，特别是对传送带下面的三栏。同样，下面还要去掉这个模板中的 row 和 column 类。

(6) 紧接着是类为 content 的 div 标签，去掉 row 类，结果如下：

```
<div class="content">
```

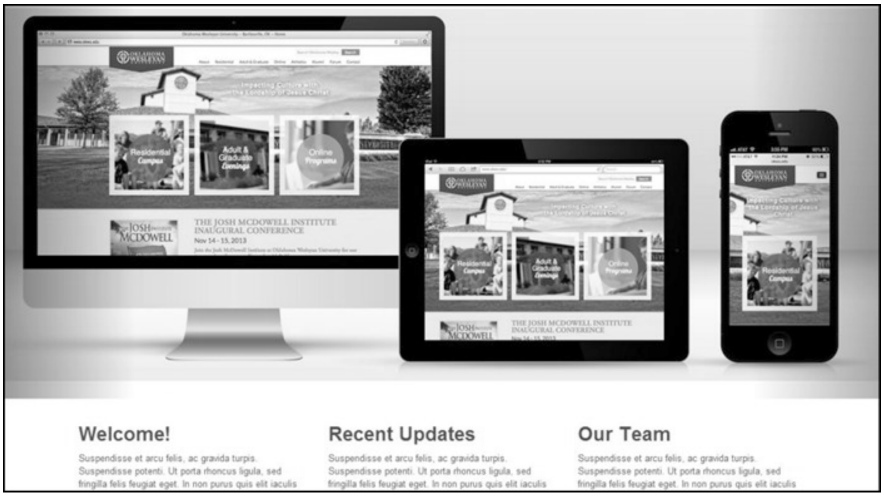
(7) 最后，还要去掉类为 main 的 div 中的 PHP 脚本，因为我们不需要 roots_main_class 在这里生成的 column 类，这行代码应该变成这样：

```
<div class="main" role="main">
```

(8) 保存文件。

(9) 在浏览器中刷新 Home 页面。

应该看到传送带已经与屏幕一样宽了。



是不是有点像变魔术？

还记得我们贴到所见即所得编辑器中的代码中包含的 container、row 和 column 类吧，它们就是用来约束页面内容宽度的。

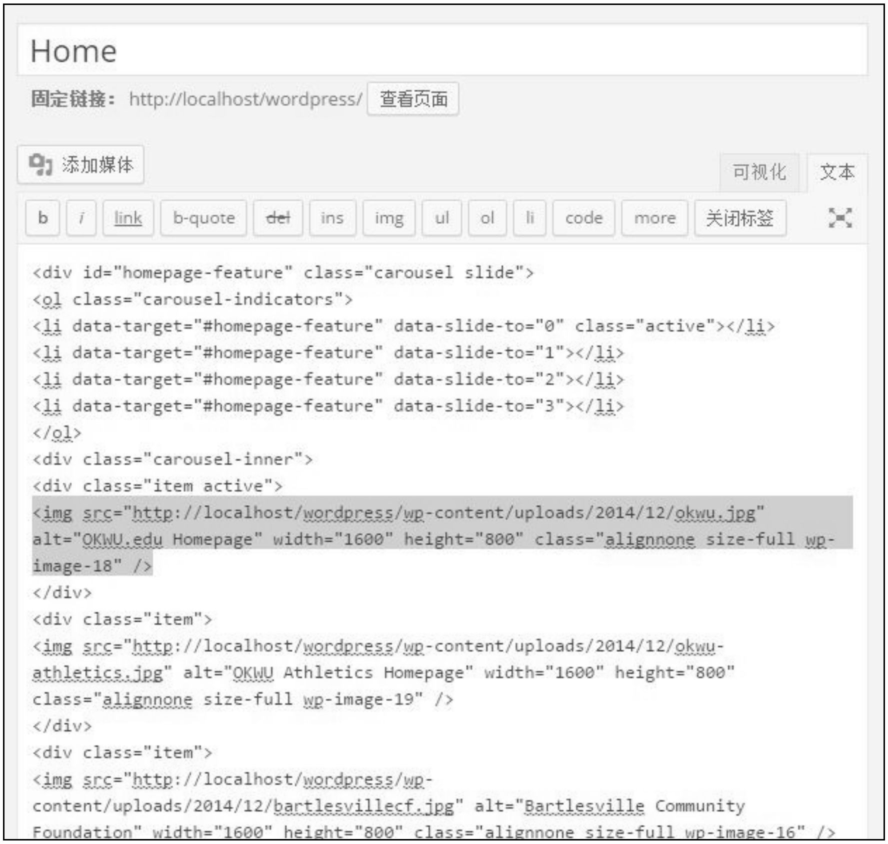
接下来差不多就可以自定义样式了。但在此之前，为了让标记更容易维护，我们得再做一些清理工作。

3.8 在自定义结构中使用自定义栏目

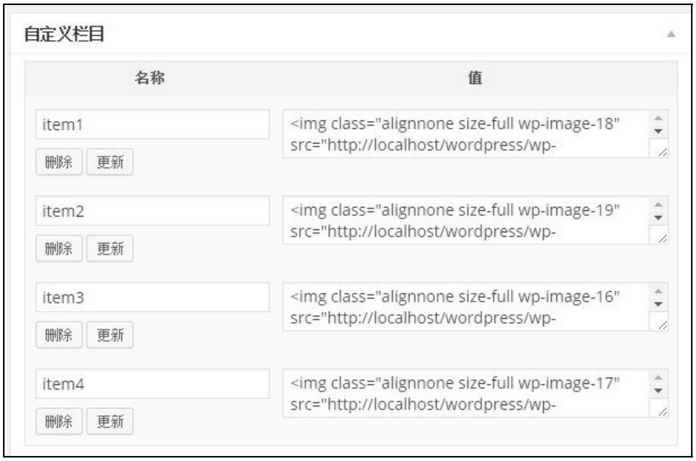
如前所述，所见即所得编辑器并不适合用来编辑我们的主页内容。因为这个页面里要自定义的标记结构很多。WordPress 内置的这个编辑器是用于编写文字和图片的，不适合容器、行、列和传送带。因此，我们还是使用 WordPress 的自定义栏目来控制这些内容吧。关于 WordPress 的自定义栏目（custome field），可以参见这里：http://codex.wordpress.org/Custom_Fields。

操作步骤简单直观。我们要为传送带中的每一张图片创建一个自定义栏目，然后再为它下面的每一栏创建一个自定义栏目。

(1) 在 WordPress 编辑器中打开 Home 页面，复制每张图片的标记：



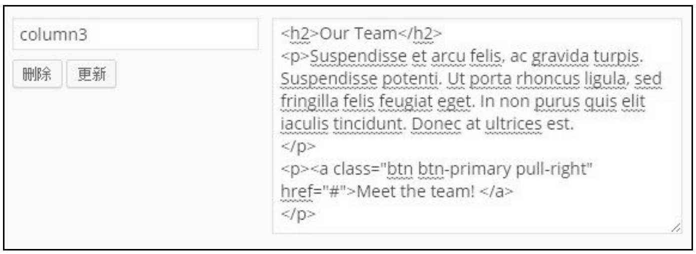
然后，分别为它们创建一个自定义栏目[①]，命名为 item1、item2、item3 和 item4。



(2) 接下来是栏内容，分别命名为 column1、column2 和 column3，包括标题、段落和按钮标记。



注意每一栏的内容并不止上图中显示的内容，拖动文本区，可以看到其中包含的全部内容：



① WordPress 3.1 之后，编辑页面中的某些选项默认会隐藏，这时请单击页面右上角的"显示选项"，然后勾选"自定义栏目"，就可以在编辑区下面看到自定义栏目了。——译者注

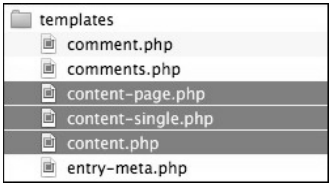
(3) 更新页面，保存我们的工作。

好了，现在该把这些栏目放到目标标记中，也就是说又要创建模板了。

3.9 创建自定义的内容模板

我们已经为主页创建了自定义的基本结构，设置了自定义的页面模板，去掉了常规的页面标题。接下来轮到为页面内容创建自定义模板了。

Roots 用来管理内容循环的模板文件是 templates 文件夹中的 content-page.php、content-single.php 和 content.php：



打开文件，可以看到标准文章和页面的循环指令。

我们要创建 content-page.php 的自定义版本。

(1) 复制 content-page.php。
(2) 把副本命名为 content-home.php。
(3) 在编辑器中打开 content-home.php，可以看到如下代码：

```
<?php while (have_posts()) : the_post(); ?>
  <?php the_content(); ?>
  <?php wp_link_pages(array('before' => '<nav class="pagination">',
    'after' =>'</nav>')); ?>
<?php endwhile; ?>
```

(4) 这个循环做了两件事：

- ❑ 从所见即所得编辑器中取得内容
- ❑ （如果需要）创建分页链接

(5) 我们想要修改循环，取得所见即所得编辑器中的自定义栏目，所以删除这一行：

```
<?php the_content(); ?>
```

(6) 主页不需要显示分页链接，因此把这一行也删除：

```
<?php wp_link_pages(array('before' => '<nav class="pagination">',
  'after' => '</nav>')); ?>
```

删除之后我们再放一句话，作为测试用。

(7) 输入：Hello this is a test!。
(8) 接下来需要修改 page-home.php 模板，使用这个新的内容循环。
(9) 打开 page-home.php。
(10) 修改下面这行代码：

```
<?php get_template_part('templates/content', 'page'); ?>
```

把“page”改成“home”：

```
<?php get_template_part('templates/content', 'home'); ?>
```

这样就可以让这个模板文件使用 templates 文件夹中新的内容模板 content-home.php 了。

(11) 保存修改。
(12) 刷新主页。

应该看到一行文本和默认的页脚。



祝贺你，新的内容模板生效了。现在该给它添加内容了，我们要通过自定义栏目来构建传送带。

3.9.1　通过自定义栏目构建传送带

我们需要从相应的自定义栏目取得每个传送带图片的标记。如前所述，关于自定义栏目的详细信息，可以看这里：http://codex.wordpress.org/Custom_Fields。

下面从取得每一项开始。

(1) 删除 content-home.php 中的测试文本，此时文件就只剩下面两行代码了：

```
<?php while (have_posts()) : the_post(); ?>
<?php endwhile; ?>
```

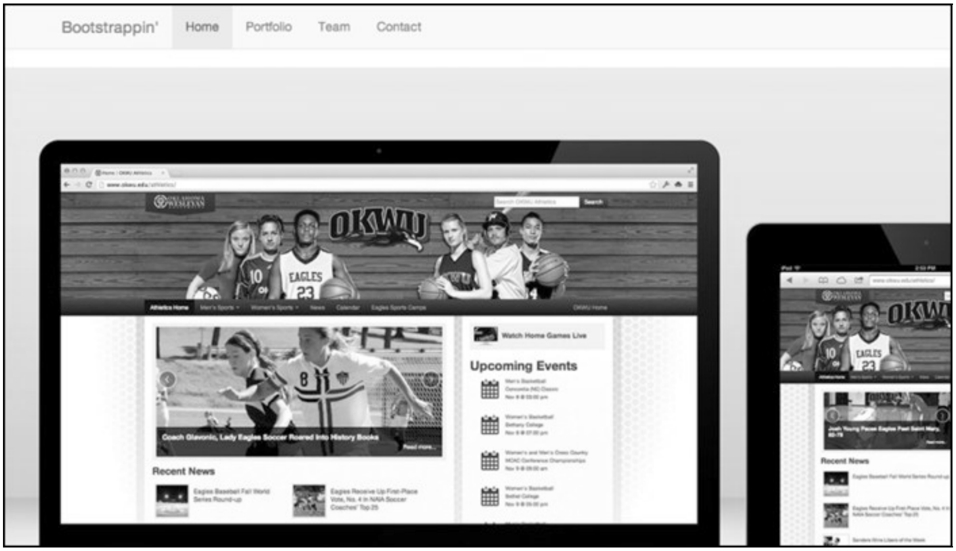
(2) 接下来我们要在这两行之间添加代码。
(3) 要取得传送带的第一项，需要引用该自定义栏目的名称 item1，那就要使用下面这行代码：

```
<?php $item="item1"; echo get_post_meta($post->ID, $item, true); ?>
```

这行代码为 item1 创建一个 PHP 变量，并在 get_post_meta()模板标签中引用了它。

参数`$post->ID`、`$item` 和 `true` 表示要取得当前文章（或页面）中名为 item1 的栏目，并返回其字符串表示。（最后一个参数如果传入 `false`，则返回一个数组。）

(4) 保存文件，刷新主页，应该看到第一张图片：



(5) 好了，下面就只要复制修改上面那行代码，分别取得对应项的标记即可，结果如下：

```
<?php $item="item1"; echo get_post_meta($post->ID, $item, true); ?>
<?php $item="item2"; echo get_post_meta($post->ID, $item, true); ?>
<?php $item="item3"; echo get_post_meta($post->ID, $item, true); ?>
<?php $item="item4"; echo get_post_meta($post->ID, $item, true); ?>
```

保存并刷新主页，应该能看到全部四张图片，从上到下显示在页面上。

接下来要把它们封装在传送带标记中。（别忘记，相应的标记可以到 index.html 或者那个所见即所得编辑器里面找。）

(6) 先复制父 `div` 及紧随其后的传送带指示器，代码片段如下：

```
<?php while (have_posts()) : the_post(); ?>
  <div id="homepage-feature" class="carousel slide">
    <ol class="carousel-indicators">
      <li data-target="#homepage-feature" data-slide-to="0"class="active"></li>
      <li data-target="#homepage-feature" data-slide-to="1"></li>
      <li data-target="#homepage-feature" data-slide-to="2"></li>
      <li data-target="#homepage-feature" data-slide-to="3"></li>
    </ol>
```

(7) 然后是 `carousel-inner` 元素的开始标签，以及封装每张图片的类为 `item` 的 `div`，第一个还要加个 `active` 类：

```
<div class="carousel-inner">
  <div class="item active">
    <?php $item="item1"; echo get_post_meta($post->ID, $item, true); ?>
  </div>
  <div class="item">
    <?php $item="item2"; echo get_post_meta($post->ID, $item, true); ?>
  </div>
  <div class="item">
    <?php $item="item3"; echo get_post_meta($post->ID, $item, true); ?>
  </div>
  <div class="item">
    <?php $item="item4"; echo get_post_meta($post->ID, $item, true); ?>
  </div>
</div><!-- /.carousel-inner -->
```

(8) 最后是传送带的控件。

```
<!-- Controls -->
  <a class="left carousel-control" href="#homepage-feature" data-slide="prev">
  <span class="icon-prev"></span>
  </a>
    <a class="right carousel-control" href="#homepage-feature" data-slide="next">
  <span class="icon-next"></span>
  </a>
</div><!-- /#homepage-feature.carousel -->
```

(9) 填充完以上传送带标记，保存文件。

刷新浏览器，就又能看到传送带了。



下一节我们添加三栏内容。

3.9.2 使用自定义栏目构建三栏内容

添加分栏内容的过程与上一节类似。

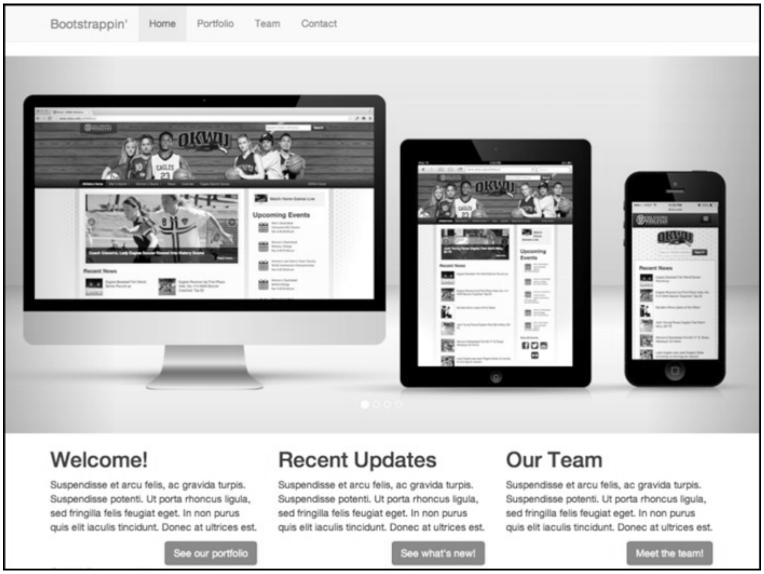
使用如下模板标签可以取得每一栏：

```
<?php $column="column1"; echo get_post_meta($post->ID, $column, true); ?>
<?php $column="column2"; echo get_post_meta($post->ID, $column, true); ?>
<?php $column="column3"; echo get_post_meta($post->ID, $column, true); ?>
```

然后，同样需要把每一栏内容封装到各自的标记中，包括相应的 container、row 和 column 类。

```
<div class="page-contents container">
   <div class="row">
      <div class="col-sm-4">
        <?php $column="column1"; echo get_post_meta($post->ID, $column, true); ?>
      </div>
      <div class="col-sm-4">
        <?php $column="column2"; echo get_post_meta($post->ID, $column, true); ?>
      </div>
      <div class="col-sm-4">
        <?php $column="column3"; echo get_post_meta($post->ID, $column, true); ?>
      </div>
   </div><!-- /.row -->
</div><!-- /.container -->
```

保存结果，刷新主页，应该能看到传送带下面出现了三栏内容。



接下来，我们再加入页脚内容。

3.10　加入页脚内容

Roots 自带了一个页脚小工具，我们可以用它来布置社交媒体图标。

(1) 在 WordPress 的仪表板中，打开“外观 | 小工具”。

(2) 在最右边，可以看到“页脚”。

(3) 单击它以扩展。

(4) 把“文本”拖进去。

(5) 从 index.html 中复制社交媒体图标部分的代码片段：

```
<ul class="social">
  <li><a href="#" title="Twitter Profile"><span class="icon fa
fa-twitter"></span></a></li>
  <li><a href="#" title="Facebook Page"><span class="icon fa
fa-facebook"></span></a></li>
  <li><a href="#" title="LinkedIn Profile"><span class="icon fa
fa-linkedin"></span></a></li>
  <li><a href="#" title="Google+ Profile"><span class="icon fa
fa-google-plus"></span></a></li>
  <li><a href="#" title="GitHub Profile"><span class="icon fa
fa-github-alt"></span></a></li>
</ul>
```

(6) 把代码片段粘贴到大文本区中。、



(7)“标题”空着。
(8) 不要选择“自动分段”复选框。
(9) 单击“保存”按钮。
(10) 刷新主页。

图标是放在一个无序列表中的。由于没有应用样式，而且也没有设置图标字段，所以只能看到无序列表的项目符号。



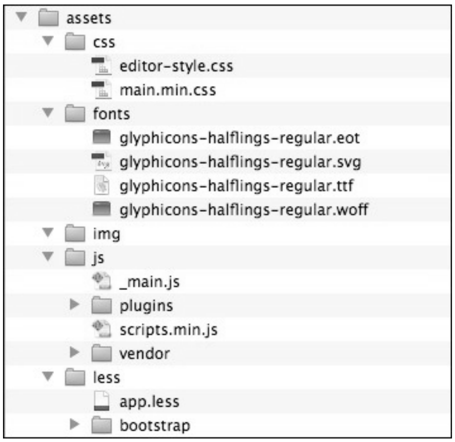
看来，下面必须得把这个自定义站点的资源用起来了。

在使用 Roots 的资源之前，我们还是先看看它提供了什么资源吧。这样我们就会对下一步要做的事情胸有成竹了。

3.11　Roots 的 assets 文件夹里有什么

Roots 主题把自己的 css、less、js 和 img 文件夹集中放在了 assets 文件夹中。打开 assets 文件夹，你会发现其中的结构与第 2 章的个人作品站点的结构非常相似，当然那又是基于 HTML5 Boilerplate 的。

稍后我们会用自己的资源来替换 Roots 的资源。但在此之前，我们应该知道 Roots 是怎么调动这些资源的。可以把 Roots 的 assets 文件夹全部展开，内容如下图所示：



Roots 的在线文档（http://roots.io/roots-101/#theme-assets）对这些文件之前如何配合给出了解释。

- less 文件夹中的 app.less 负责管理所有样式规则。这个文件先从 bootstrap 文件夹中的 bootstrap.less 文件中引入所有 Bootstrap 样式，然后给出了很多注释和推荐的选择符，在此基础上可以编写自定义的 LESS 代码。
- 这个 app.less 文件会被编译成 main.min.css 文件保存到 css 文件夹中。
- 同样在 css 文件夹中，还有一个 editor-style.css 文件，是用来自定义 WordPress 可视化编辑器外观的。
- js 文件夹中包含_main.js 文件，是用来保存自定义 JavaScript 代码的。（这个文件中定义了一个方法，必要时可用于指定和限制其中 JavaScript 代码的作用域。）
- js 文件夹中还有一个 plugins 文件夹，用于保存 Bootstrap 插件，也可用来管理其他插件。
- 与 HTML5 Boilerplate 类似，vendor 文件夹中包含的基本上都是独立的文件库，默认包含 jQuery 和 Modernizr。
- 另外，Roots 的设计不是把所有插件组合到一个 plugins.js 文件中，而是通过_main.js 整合所有插件，并生成一个叫 scripts.min.js 的文件。

在 Roots 主题的根文件夹中，还有一个 Gruntfile.js 文件，可以通过它把 LESS 编译成 CSS，以及合并和压缩 JavaScript 文件（系统中必须安装 Grunt）。

当然，我们不依靠 Grunt 帮我们做这些。前几章使用的 LESS 编译器就够用了。至于合并压缩 JavaScript，我们也采用其他方式，具体可参见附录 A。

目前，我们只想把第 2 章弄好的样式、脚本和字体都拿进来。而最简单、最直观的方式就是用我们自己的 assets 文件夹来替换 Roots 的 assets 文件夹。

3.12　更换设计资源

在本章的练习文件中，有一个__BOOTSTRAPPIN_PORTFOLIO_ASSETS 文件夹，其中有第 2 章的资源文件，不过稍微有点改动。

要是你真的去看，会发现改动了这么几个地方。首先我把 favicon.ico 和 apple-touch-icon-precomposed.png 文件从主文件夹挪到了 ico 文件夹中。但最关键的还是我按照附录 A 的操作步骤优化压缩了 CSS 和 JavaScript 文件：main.css、main.js 和 plugins.js。

打开__BOOTSTRAPPIN_PORTFOLIO_ASSETS 文件夹，可以看到下图所示的结构，其中会直接在我们主题中使用的都被选中突出显示了：



最容易的推进方式就是直接替换：

(1) 把 Roots 的 assets 文件夹重命名为__ROOTS-ASSETS-ORIGINAL；
(2) 重命名__BOOTSTRAPPIN_PORTFOLIO_ASSETS 为 assets；
(3) 接下来别忘了更新 CSS 和 JavaScript 文件的链接，因为 Roots 的命名方式不太一样；
(4) 如果现在就刷新站点，会发现一切都乱套了，当然，这是没有应用样式表的结果。

下面我们来链接样式表。

3.13　链接样式表

链接样式表就是更新 Roots 文件，使用 main.css 样式表。

(1) 查看 WordPress 的源代码，会发现样式表链接的路径没错，但文件名是后缀有版本号的 main.min.css，以我本地安装的 WordPress 为例，样式表的路径如下：

```
<link rel="stylesheet" href="http://localhost:8888/bootstrappin-portfolio/
assets/css/main.min.css?ver=9a2dd99b82ca338b034e8730b94139d2">
```

> Roots Gruntfile 会使用 MD5 散列算法生成版本号。（参见在线文档：http://roots.io/using-grunt-for-wordpress-theme-development/。）这种工作流程很好，但超出了本书讨论范围。我们不会给文件生成版本号。

(2) 只要把链接中的文件名改成 main.css 就行了。
(3) Roots 通过 lib 文件夹中的 scripts.php 来管理到样式表和脚本文件的链接。
(4) 用编辑器打开 scripts.php。

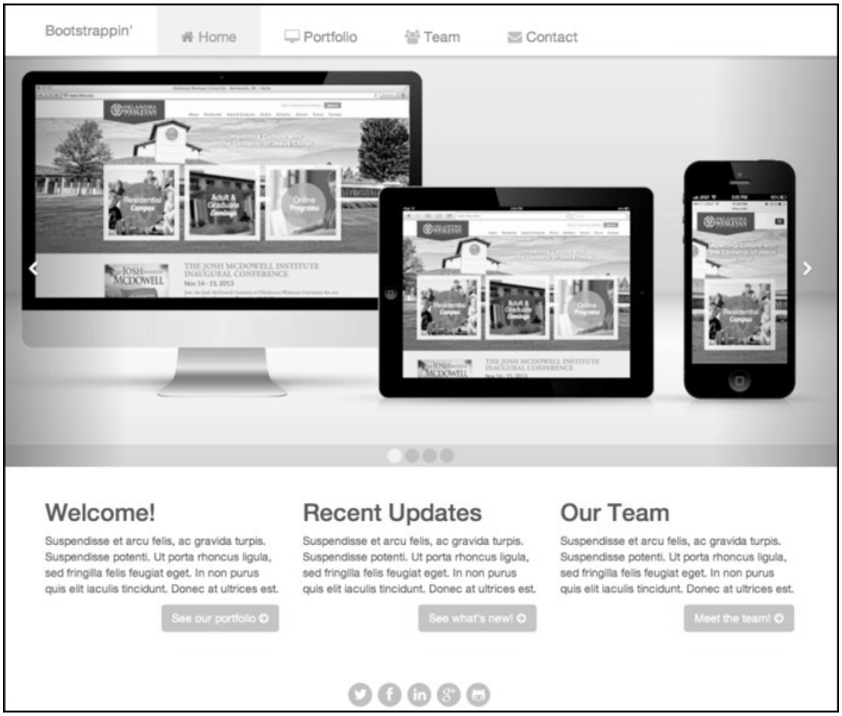
(5) 找到现在这行代码：

```
wp_enqueue_style('roots_main', get_template_directory_uri()
  . '/assets/css/main.min.css', false, '9a2dd99b82ca338b034e8730b94139d2');
```

把它改成：

```
wp_enqueue_style('roots_main', get_template_directory_uri()
  . '/assets/css/main.css', false, null);
```

(6) 保存修改，刷新浏览器看看效果。应该可以看到我们自定义的样式，还有 Font Awesome 字体图标，如下所示：



休息一会儿。

此时传送带并不会运行，而且响应式的导航按钮也不起作用。这些都需要重新链接 JavaScript 文件。

3.14 链接 JavaScript 文件

如前所述，Roots 用同一个文件管理 JavaScript 和 CSS 文件的链接。

好，现在 scripts.php 还在编辑器中打开着，首先来检查一下 jQuery 的本地后备仍然有效。

(1) 找到以下代码，大概在 scripts.php 文件的中间位置：

```
if ($add_jquery_fallback) {
  echo '<script>window.jQuery || document.write(\'<script
    src="' . get_template_directory_uri() .
    '/assets/js/vendor/jquery-
    1.10.2.min.js"><\/script>\')</script>' . "\n";
  $add_jquery_fallback = false;
}
```

(2) 需要确认这里的路径和文件名与下面一致：

```
/assets/js/vendor/jquery-1.10.2.min.js
```

在本书写作时，练习文件夹 03_Code_BEGIN 中保存的第 2 章所用的 HTML5 Boilerplate 版本与 Roots 主题是同步的，所以文件一致。

如果你想使用不同版本的 jQuery，还应该更新 Google CDN 的链接，相应代码在稍靠前的位置：

```
if (!is_admin() && current_theme_supports('jquery-cdn')) {
  wp_deregister_script('jquery');
  wp_register_script('jquery',
    '//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js',
    false, null, false);
```


(3) 接下来再检查 Modernizr 脚本的链接。

> 第 1 章介绍过，Modernizr 可以让 Internet Explorer 8 支持 HTML5，以及其他一些新特性。所以我们确实需要它。

在文件上方 `roots_scripts()` 函数中，可以找到 Modernizr 的链接：

```
wp_register_script('modernizr', get_template_directory_uri() .
  '/assets/js/vendor/modernizr-2.6.2.min.js', false, null,
  false);
```

这里要看一下路径、文件名和版本是否匹配。同样，本书提供的练习文件是匹配的。如果你愿意，可以自己改。

(4) 最后，我们得添加到 plugins.js 和 main.js 文件的链接，切断原来 Roots 模板中的文件链接。我们先删除，后添加。

- 打开 lib 文件夹的 scripts.php 文件。
- Roots 是把插件和自定义脚本组合到一个文件中，并使用以下代码注册的：

```
wp_register_script('roots_scripts',
  get_template_directory_uri() .
  '/assets/js/scripts.min.js', false,
  '2a3e700c4c6e3d70a95b00241a845695', true);
```

把这一行删除或注释掉。

- 接着还要删除下面这行，就在前面那行代码后面：

```
wp_enqueue_script('roots_scripts');
```

- 接着用如下代码注册我们的两个脚本文件：

```
wp_register_script('plugins_script',
  get_template_directory_uri() . '/assets/js/plugins.js',
  false, null, true);
wp_register_script('main_script',
  get_template_directory_uri() . '/assets/js/main.js',
  false, null, true);
```

- 再把它们加入队列：

```
wp_enqueue_script('plugins_script');
wp_enqueue_script('main_script');
```

- 保存并刷新浏览器。

查看源代码，应该可以看到下列这些行位于`</footer>`标签和`</body>`标签之间（完整的 URL 可能会与你的有所不同）：

```
</footer>
<script type='text/javascript'
  src='http://localhost:8888/bootstrappin-
  portfolio/assets/js/plugins.js'></script>
<script type='text/javascript'
  src='http://localhost:8888/bootstrappin-
  portfolio/assets/js/main.js'></script>
</body>
</html>
```

测试传送带，应该正常运行了。

测试响应式导航条，在窄屏幕下也应该可以折叠并出现下拉按钮，而且单击按钮可以扩展和折叠！

下一节，我们为导航条和页脚添加 Logo 图片。

3.15　为导航条和页脚添加 Logo 图片

先把盛 Logo 图片的标记放到 `navbar-brand` 链接里面。相应的标记位于 templates 文件夹中的 header-top-navbar.php 文件内。

(1) 用编辑器打开 templates 文件夹中的 header-top-navbar.php 文件。
(2) 找到下列元素：

```
<a class="navbar-brand" ...
```

(3) 删除以下标签，它的用处是在导航条品牌链接中插入站点名称：

```
<?php bloginfo('name'); ?>
```

(4) 在刚刚删除上一行的位置，填入盛放 Logo 的标签：

```
<img src="<?php echo get_template_directory_uri(); ?>/assets/img/
logo.png" width="120" alt="Bootstrappin'">
```

> 别忘了咱们的 Logo 图片做得很大，因为想适应视网膜屏嘛。所以这里一定要加上 width 属性，否则就太大了。

(5) 保存结果。

(6) 刷新页面，应该可以看到 Logo 图片，如下图所示：



3

接下来是页脚。

页脚中的社交媒体图标应该没问题。Roots 主题默认的版权信息显示在这些图标下方。为了练习起见，我们删除这个版权信息，然后把站点的 Logo 放到社交媒体图标上方。

为此，我们需要编辑页脚的模板文件。

(1) 在编辑器中打开 template 文件夹中的 footer.php 文件。

(2) 打开后删除下列代码，我们的练习文件不需要声明版权：

```
<p>&copy; <?php echo date('Y'); ?> <?php bloginfo('name');
  ?></p>
```

(3) 然后在动态的侧边栏上面插入一行：

```
[新插入的行]
<?php dynamic_sidebar('sidebar-footer'); ?>
```

(4) 添加一个带链接的站点 Logo：

```
<p><a href="<?php echo home_url(); ?>/"><img src="<?php
  echo get_template_directory_uri();
  ?>/assets/img/logo.png" width="80"
  alt="Bootstrappin'"></a></p>
```

> 跟前面一样，别忘了这里也要加一个 width 属性；否则，图片就太大了。

这里是把 WordPress 的 home_url 模板标签包含在一个段落里。如果觉得不合适，你可以自己换成其他标签。

(5) 保存文件，刷新浏览器，应该看到类似如下的结果：

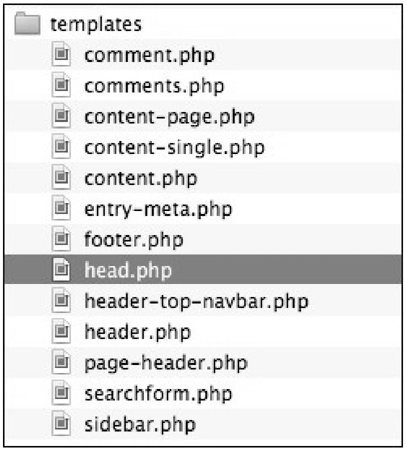


我们的网站设计至此基本完成了——但是别忘了还有站点的收藏图标和触摸设备图标啊。

3.16　添加图标链接

为了让我们的主题能够跨设备，还需要在 head.php 模板文件中添加到 favicon.ico 和 apple-touch-icon-precomposed.png 文件的链接。

(1) 在编辑器中打开 templates 文件夹中的 head.php 文件。



(2) 用 WordPress 的 PHP 函数 get_template_directory_uri()取得路径的前半部分，然后补上 favicon.ico 和 apple-touch-icon-precomposed.png 文件的具体路径:

```
<!-- Icons -->
<link rel="shortcut icon" href="<?php echo
  get_template_directory_uri(); ?>/assets/ico/favicon.ico">
<link rel="apple-touch-icon-precomposed" href="<?php
  echo get_template_directory_uri(); ?>/assets/ico/apple-
  touch-icon-precomposed.png">
```

(3) 保存并刷新浏览器，应该可以看到收藏图标出现在标签页的标题栏上。

以下是在 Chrome 浏览器中看到的图标：



最后，我们还要关注两个细节，以满足 WordPress 主题的特殊需求。

3.17 恢复 WordPress 特有的样式

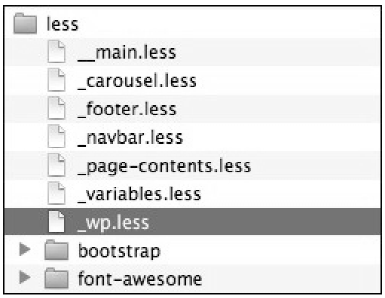
正是考虑到要恢复 WordPress 特有的样式，所以我们之前才把 Roots 的资源文件夹重命名为__ROOTS_ASSETS_ORIGINAL。虽然我们的主题已经完整了，但 Roots 中却包含了两组重要的 WordPress 样式，还必须把它们请回来。

首先，在 Roots 的 css 文件夹中，可以看到 editor-style.css 文件。这个文件用于改进编辑器的使用体验，我们要把它复制到我们主题的 css 文件夹中。（读者也可以自己创建一个类似的文件，使其与自己站点的风格一致。）

3

其次，如果我们想在站点上架一个博客，或者想发布一款主题，那么就应该恢复一些 WordPress 特有的样式。这些样式 Roots 都放到了 less 文件夹下的 app.less 文件中。把这个文件添加到自定义的 LESS 文件，然后重新编译到 main.css 文件中只是小菜一碟。具体步骤如下。

(1) 在编辑器中打开 Roots 主题 less 文件夹下的 app.less 文件。
(2) 在另一个编辑器窗口中，创建一个新文件，命名为_wp.less 并保存到我们自定义的 assets 文件夹下的 less 文件夹下，结果如下图所示：



(3) 从 app.less 中复制如下代码，保存到_wp.less 中，这些代码针对 WordPress 为图片生成的类：

```
/* ========================
   WordPress Generated Classes
   ...
   ======================== */

aligncenter { display: block; margin: 0 auto; }
alignleft { float: left; }
alignright { float: right; }
figure.alignnone { margin-left: 0; margin-right: 0; }
```

(4) 如果你想在搭建一个博客，那么为了加快设计过程，还应该把下面这些针对文章的选择符复制过去。这些选择符涵盖了 Roots 模板针对博客文章的元素和类名：

```
/* ========================
   Posts
   ======================== */

hentry header { }
hentry time { }
hentry .byline { }
hentry .entry-content { }
hentry footer { }
```

(5) 如果你想在站点中使用侧边栏，也要提取出`.sidebar`选择符。

(6) 可能还要提取 Roots 默认使用的其他一些选择符，如`.content`、`.main`、`.sidebar`，等等。

(7) 还有针对画廊短码的样式：

```
/* Gallery Shortcode */
.gallery-row { padding: 15px 0; }
```

(8) 复制完需要保留的代码之后，再花点时间把块注释改成单行注释，以便它们不会被编译到 CSS 文件中。

```
//     Posts
//     ------------------------
...

//     WordPress Generated Classes
//     ------------------------
```

(9) 保存_wp.less。

(10) 关闭 app.less。

(11) 现在打开__main.less，添加一行代码导入_wp.less:

```
 // Other custom files
 @import "_page-contents.less";
 @import "_footer.less";
 @import "_wp.less";
```

(12) 重新编译__main.less 为 css/main.css，确认选择了最小化输出，以保证最佳性能。

全部搞定！我们不仅集成了自己的设计，这个主题还可以随时基于 WordPress 进行二次开发。

3.18　小结

下面来回顾一下本章内容。

- 本章以杰出的 Roots 主题作为我们 WordPress 主题的起点。
- 我们自定义了如下模板文件，修改了标记结构。
  - head.php：用于添加收藏图标和触摸设备图标的链接。
  - header-top-navbar.php：用于添加 Logo 图片。
  - footer.php：用于添加 Logo 图片。
- 我们新建了如下自定义模板文件。
  - page-home.php，基于 template-custom.php。
  - base-page-home.php，基于 base.php。
  - content-home.php，基于 content-page.php。
- 对于复杂的主页内容，我们利用了 WordPress 的自定义栏目。
- 为了放置社交媒体图标，我们使用了一个页脚部件。
- 我们整合了自己的编译资源，包括 LESS、CSS 和 JavaScript。
- 为整合自定义的设计资源，我们编辑了 Roots 的 scripts.php 文件，更新了指向我们 CSS 和 JavaScript 文件的链接。
- 我们还从 Roots 中提取了一组样式，用于匹配 WordPress 站点的细节。

你能跟着一步一步做完，真了不起！

本章介绍的流程适用于将任意 Bootstrap 设计转化为 WordPress 主题。

好了，我们还是回到 Bootstrap。下一章，我们讨论如何设计一个企业网站。

4 企业网站

上一章把个人作品站点改造成了 WordPress 主题。本章，轮到我们充实这个作品站点，补充一些项目，从而展示我们的能力了。换句话说，我们要构建一个复杂的企业主页。

请大家花点时间看一看下面几家成功企业的网站：

- Zappos（http://zappos.com）
- Amazon（http://amazon.com）
- Adobe（http://adobe.com）
- HP（http://hp.com）

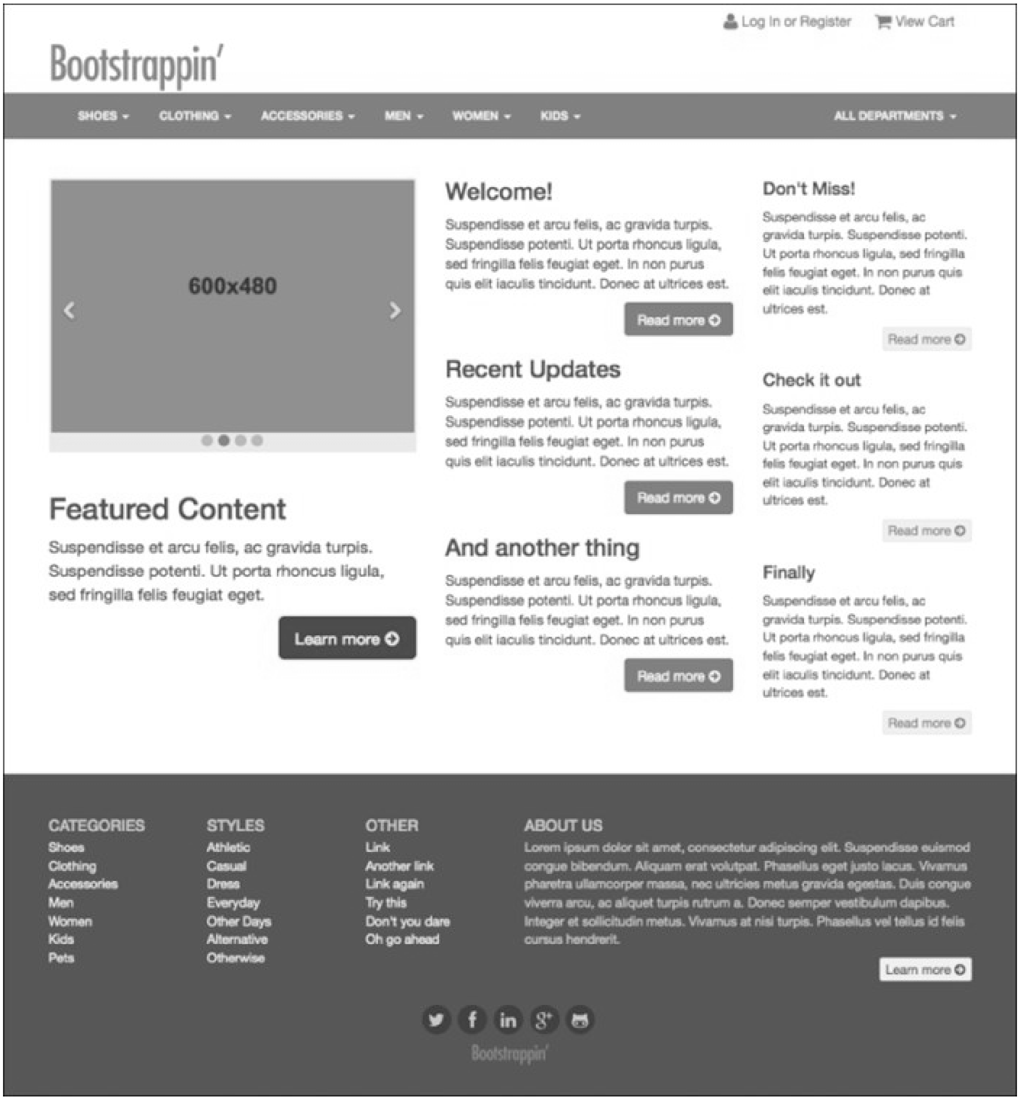
尽管这些网站各有特色，但共同的一点就是它们都很复杂。

如果按照区域划分，可以将这些网站的主页分成三部分。

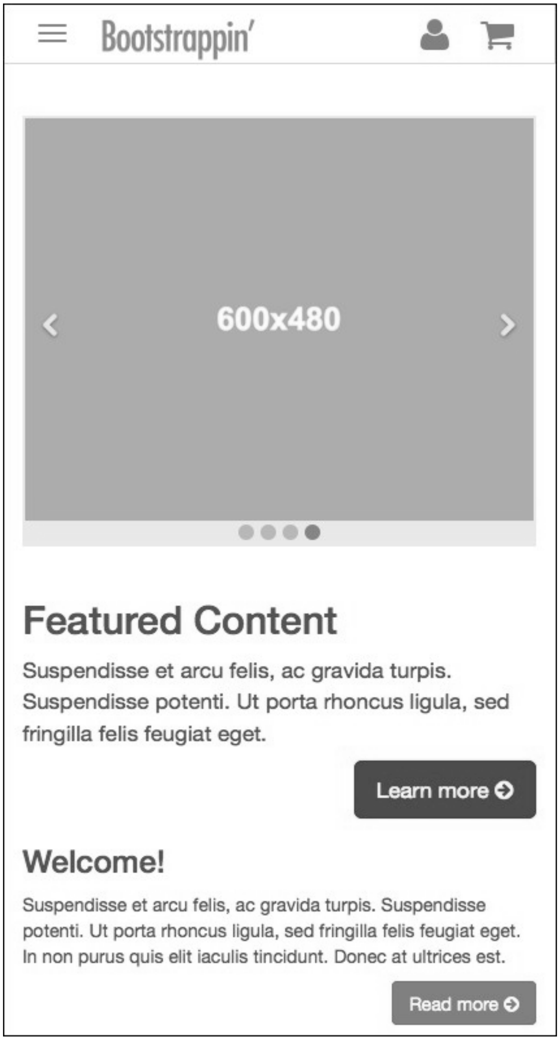
- **页头区**：这一部分包含 Logo、带下拉菜单的主导航、二级和实用链接导航，以及登录和注册选项。
- **主内容区**：这一部分布局复杂，至少三栏。
- **页脚区**：包含多栏链接和信息。

我们必须能够掌控这些复杂性。为此，需要充分利用 Bootstrap 的 12 栏响应式网格系统。

以下是我们打算要构造的设计在中、宽视口中的效果：



在窄视口中，页面会相应变化，如下图所示：



这样，我们需要做以下这些事。

(1) 以第 2 章的个人作品网站作为起点。
(2) 完成复杂的页头区，包括 Logo、导航以及右上角的实用导航（桌面视口）。
(3) 在较窄的视口中，实用导航只显示为图标，与折叠后的响应式导航条并列。
(4) 要实现企业风格的配色方案。
(5) 调整桌面版和响应式导航条。
(6) 为主内容区和页脚区设置复杂的多栏布局。

先做最核心的工作吧——准备项目的启动文件。

4.1 准备启动文件

与本书中的其他项目一样，本章的启动文件也可以从 Packt Publishing 网站下载：http://www.packtpub.com/support。下载后解压缩，找到文件夹 04_Code_BEGIN 即可。

这些文件基本上源自第 2 章，因此已经具备了以下重要组件。

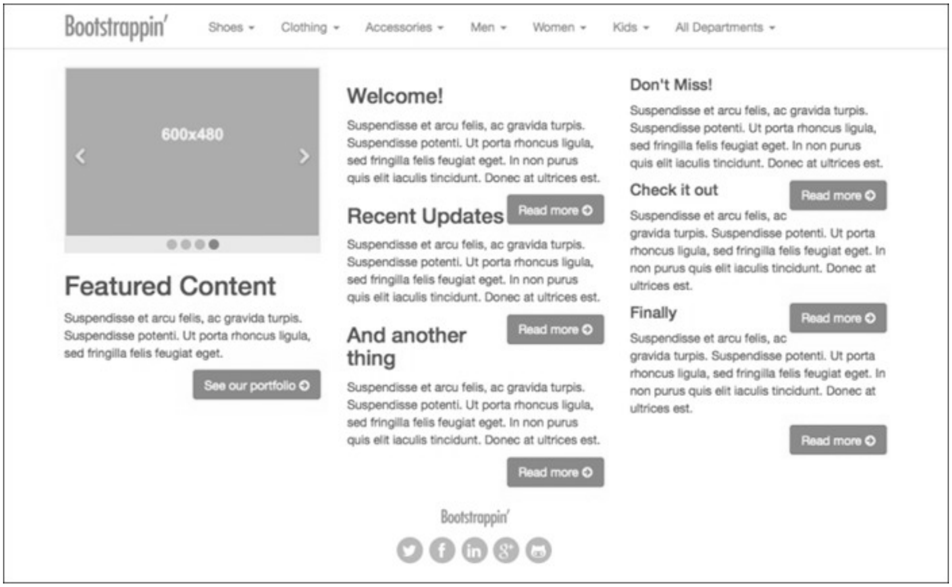
- Bootstrap LESS 和 JavaScript 文件，分别位于下列文件夹：
  - less/bootstrap：Bootstrap 的 LESS 文件
  - js/bootstrap：Bootstrap 的个别插件
  - js/plugins.js：Bootstrap 压缩后的插件
- HTML5 Boilerplate 及下列文件：
  - 基本的结构化标记文件 index.html
  - js/vendor/modernizr-2.6.2.min.js
  - js/vendor/query-1.10.2.min.js
- 保证兼容 Internet Explorer 8 的 repond.js：
  - js/vendor/respond.js
- Font Awesome 字体图标，包括：
  - fonts 文件夹中的图标字体
  - less/font-awesome 文件夹中的 LESS 文件

除了以上重要的资源之外，我们还在构造第 2 章的网站时添加过一些 LESS 文件，可以在 less 文件夹中找到它们。

- __main.less：基于 bootstrap.less,，导入了位于 less/bootstrap 中的 Bootstrap 的 LESS 文件、Font Awesome 字体图标和我们自定义的 LESS 文件。
- _carousel.less：基于 Bootstrap 的 carousel.less，自定义了传送带的内边距、背景和指示图标。
- _footer.less：包含 Logo 及社交媒体图标的布局和设计样式。
- _navbar.less：基于 Bootstrap 的 navbar.less，调整了 `.navbar-brand` 类的样式，以使导航条中的 Logo 位置合适。
- _page-contents.less：其中的样式确保了每一栏中的浮动按钮在呈现为单栏的情况下相互清除。
- _variables.less：基于 Bootstrap 和 variables.less，针对导航条和传送带自定义了灰颜色、新增了变量。

> 如果你愿意，也可以把全新的 Bootstrap 作为本章练习的起点。只要把 Font Awesome 图标换成 Glyphicons 即可。虽然这样就没有前面所说的样式了，但你可以自己随意添加和调整。
>
> 在本章提供的启动文件中，我已经尽力在自定义的地方添加了单行注释，比如`// edited`或`// added`。

接下来，我们看一看 index.html 文件提供的内容，我已经调整了这个页面。在浏览器中打开它，应该看到如下界面：



下面说明一下相关的特性。

- 导航条很复杂，有 7 项，每一项都有下拉菜单。
- 三栏中的第一栏开头是一个传送带，后面是一个标题、一个段落和一个按钮。
- 第二和第三栏同样都包含标题和段落，以及“Read more ->”按钮。
- 页脚包含 Logo 和社交媒体图标。

我们必须重新组织第 2 章项目文件中的元素。传送带已经变小了，宽度由包含它的栏限制。除此之外，标记没有本质变化。

相对比较难一点的地方是这里使用了一个 JavaScript 插件 holder.js，目的是为传送带动态生成占位图片。查看源代码，就会发现在页面底部 plugins.js 插件之前，我们包含了这个 holder.js 脚本：

```
<!-- Holder.js for project development only -->
<script src="js/vendor/holder.js"></script>
```

最终的站点中是不使用占位图片的，因此单独给它注释出来很有必要。

加载了 holder.js 之后，就可以方便地把 holder.js 作为任意图片的来源。然后使用伪 URL 指定大小、颜色和填充文本，比如：

```
<img src="holder.js/600x480/auto/vine/textmode:literal" alt="Holder Image">
```

更多信息可以参考 holder.js 的文档：https://github.com/imsky/holder。

有了这些元素，以及 Bootstrap 内置的样式和行为，我们的起点就非常高了。下面我们就来实现所有特性。

首先，我们重新定位导航条，实现页头区的设计。

4.2　页头区

下面我们就从上到下，先来实现复杂的页头区，包括如下特性。

- 在桌面浏览器及较大视口中，让站点 Logo 显示在导航条之上。
- 包含菜单项的导航条，每个菜单项又都包含下拉菜单。
- 实用导航区。
- 带用户名和密码字段的登录表单。
- 注册选项。

以下是桌面浏览器中的目标结果：



窄视口中的目标结果如下：



我们先从放置站点 Logo 开始。

4.2.1　把Logo放到导航条上方

在这个设计方案里， Logo 可能出现在两个地方，视情况而定：

- ❑ 在桌面和宽屏幕中，显示在导航条上方；
- ❑ 在平板和手机屏幕中，显示在响应式导航条内部。

利用 Bootstrap 的响应式实用类，这两点我们都可以做到！方法如下。

(1) 在编辑器中打开 index.html。

(2) 找到导航条，复制 `navbar-brand` 的链接与图片：

```
<a class="navbar-brand" href="index.html"><img src=
  "img/logo.png" alt="Bootstrappin'" width="120"></a>
```

(3) 然后粘贴到导航条上方，在`<header role="banner">`标签和`<nav role= "navigation" class="navbar navbar-default">`标签之间。

(4) 把这个 Logo 用`<div class="container">...</div>`包装起来，使其被限制在 Bootstrap 居中的网格内部。

(5) 编辑 Logo 的链接，将其类名由 `navbar-brand` 改为 `banner-brand`。然后把图片宽度改为 180。

> Logo 图片其实很大，有 900px 宽。为了让它在视网膜屏上显示清晰，我们已经使用 `width` 属性（使用 CSS 规则也行）把它缩小到了 120px 宽，以便提供足够的像素密度。

结果代码如下所示：

```
<header role="banner">
  <div class="container">
    <a class="banner-brand" href="index.html"><img src="img/logo.png" alt=
      "Bootstrappin'" width="180"></a>
  </div><!-- /.container -->
  <nav role="navigation" class="navbar navbar-default">
```

保存修改，然后在浏览器中刷新页面。应该在导航条上面看到新的 Logo。



下面我们来调整 Logo，让它只在必要的时候显示。

在_variables.less 中，确认变量`@grid-float-breakpoint` 的值。可以通过搜索找到这个变量，less/bootstrap/variables.less 文件中这个变量的默认值如下：

```
// Point at which the navbar stops collapsing
@grid-float-breakpoint:     @screen-sm-min;
```

这个变量决定了导航条在窄视口中折叠，在宽视口中展开。在我们的设计中，考虑到导航的复杂性，需要在接近的下一个较宽的断点折叠导航条。因此，需要把变量的值设置为`@screen-md-min`。如果你使用的是 04_Code_BEGIN 中的文件，那么这个变量应该已经就这么设置好了（如果不是，那就照下面这样改过来）：

```
// Point at which the navbar stops collapsing
@grid-float-breakpoint:     @screen-md-min;
```

设置完这个变量后，我们要考虑让 `banner-brand` 只在中、大型视口中显示，而让 `navbar-brand` 只在小和超小型视口中显示。Bootstrap 为此提供了一组响应式实用类，具体可以参考在线文档：http://getbootstrap.com/css/#responsive-utilities。

下面我们就按照需求来应用这些类。

(1) 把 `visible-md visible-lg` 添加到 `banner-brand` 类后面：

```
<a class="banner-brand visible-md visible-lg"
  href="index.html"><img src="img/logo.png"
  alt="Bootstrappin'" width="180"></a>
```

(2) 把 `visible-xs visible-sm` 添加到 `navbar-brand` 类后面：

```
<a class="navbar-brand visible-xs visible-sm"
  href="index.html"><img src="img/logo.png"
  alt="Bootstrappin'" width="120"></a>
```

保存修改，刷新网页。在中、大型视口中，只会显示 `banner-brand` 中的 Logo：



在小型和超小型视口中，只会显示 `navbar-brand` 中的 Logo：



Bootstrap 太棒啦！

> 要是你觉得弄两套标记交替显示很乱，也可以考虑使用 JavaScript 来做这个事。
>
> 不过，目前这种做法的好处恰恰在于它不依赖 JavaScript。而且它也不会影响页面加载时间，因为两个地方引用同一张图片不需要发送新请求。归根结底，现在这种做法很可靠，也很合理，当然更容易实现。

下面我们来调整导航条。

4.2.2　调整导航条

现在的导航条包含 7 项，每项又各有子菜单，体现了一个大型复杂网站的需求。

其中下拉菜单的标记直接取自 Bootstrap 的导航条文档：http://getbootstrap.com/components/#navbar。

如果你查看结果标记，会发现以下特殊的类和属性：

- 父级 `li` 元素中的 `class="dropdown"`；
- 链接中的 `class="dropdown-toggle"`；
- 链接中的 `attribute="data-toggle"`；
- 子菜单 `ul` 中的 `class="dropdown-menu"`。

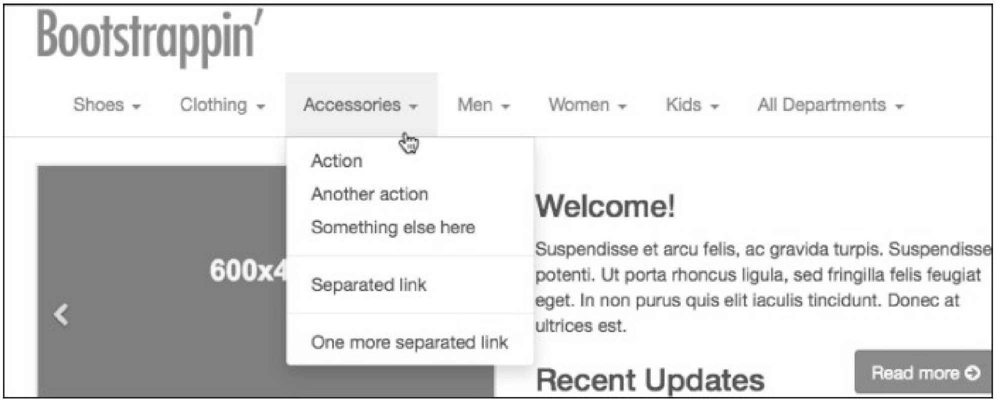
以下是结果标记：

```
<li class="dropdown">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">Shoes
      <b class="caret"></b></a>
    <ul class="dropdown-menu">
      <li><a href="#">Action</a></li>
      <li><a href="#">Another action</a></li>
      <li><a href="#">Something else here</a></li>
      ...
    </ul>
</li>
```

另外还有一个带特殊类的特殊标签，用于显示下拉菜单指示图标：`<b class="caret"></b>`。（创建这个指示图标的 CSS 在 less/bootstrap/dropdowns.less 中。）

> 假如你使用的不是本章的启动文件（04_Code_BEGIN），需要认真检查一下_main.less 中导入其他 LESS 文件的代码，确保导入了 bootstrap/dropdowns.less。此外还应该检查 plugins.js，确保其中包含了 bootstrap-dropdown.js 插件。

在 LESS、JavaScript 和标记就位的情况下，导航条及其下拉菜单应该像下面屏幕截图中一样。（注意，Bootstrap 的所有下拉菜单都要通过单击激活。）



熟悉了标记结构，并且确认菜单工作正常之后，接下来我们要把 All Departments 菜单挪到导航条的最右端，让它与其他菜单项保持最大距离。

为此，需要把相应的列表项嵌套在它自己的无序列表中。

(1) 在 All Departments 列表项之前，关闭 `ul class="nav"`这个 `ul` 标签，用于包含之前的所有列表项。

(2) 在 All Departments 列表项之前，再新建一个 `ul` 标签，类名为 `nav` 和 `navbar-nav`。添加了这个开始标签后，这个独立的列表项就具备了标准的导航菜单结构。

(3) 除了 `nav` 和 `navbar-nav` 类之外，再添加一个 `pull-right` 类，这是 Bootstrap 的一个实用类，用于把元素浮动到右侧。

以下代码中加粗的部分是新添加的代码，紧随其后的是原来的列表项和链接：

```
</ul>
<ul class="nav navbar-nav pull-right">
  <li class="dropdown">
  <a href="#" class="dropdown-toggle" data-toggle="dropdown">All
    Departments <b class="caret"></b></a>
```

保存修改并刷新页面，应该可以看到 All Departments 下拉菜单已经浮动到了导航条的最右端：



一切顺利！下面来添加实用导航。

4.3　添加实用导航

我们的设计需要提供几个实用的导航链接，让用户可以登录、注册和查看购物车。

在中大型的视口中，我们把它们放到页头区的右上角，如下图所示：



在较小的屏幕中，则把对应的链接图标显示在折叠后的导航条的最右端，如下图所示：



说做就做。

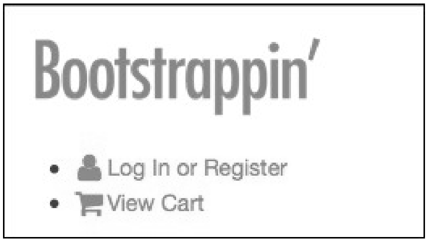
还是在 index.html 中，我们要在页头区添加实用导航项的标记，放在 `banner-brand` 元素后面。以下是完整的标记，开头是 `header` 标签，新增的代码加粗了：

```
<header role="banner">
  <div class="container">
    <a class="banner-brand visible-md visible-lg" href=
      "index.html"><img src="img/logo.png" alt="Bootstrappin'"
      width="180"></a>
    <div class="utility-nav">
      <ul>
        <li><a href="#" title="Login or Register"><i class="icon
          fa fa-user fa-lg"></i> Log In or Register</a></li>
        <li><a href="#" title="View Cart"><i class="icon fa fa-
          shopping-cart fa-lg"></i> View Cart</a></li>
      </ul>
    </div><!-- /.utility-nav -->
  </div><!-- /.container -->
```

关于以上标记，还要说明两点。

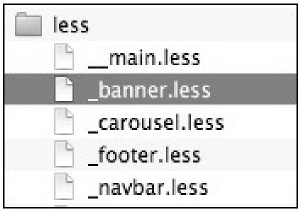
- 类 `utility-nav` 只是为了方便使用，它不是 Bootstrap 特有的类，也没有什么样式。
- 这里已经通过 `fa-user` 和 `fa-shopping-cart` 类添加了 Font Awesome 的用户和购物车图标，并通过 `fa-lg` 类把它们的尺寸增大了 33%。关于增大 Font Awesome 图标的详细说明，请参见它的文档：http://fontawesome.io/examples/#larger。

保存修改并刷新页面，应该看到新添加的 `utility-nav` 出现在了 `banner-brand` Logo 下方，如下图所示：



接下来，我们对布局进行相对位置的调整，也就是要应用一些自定义的样式。为此，要针对页头区新建一个文件，步骤如下。

(1) 创建一个新文件并命名为_banner.less，保存到 less 文件夹中，与其他自定义的 LESS 文件放在一起。



(2) 把_banner.less 添加到__main.less 的导入文件列表中。

```
// Other custom files
@import "_banner.less"; // added
```

(3) 在_banner.less 中，先给出一个帮助性的注释。把`.utility-nav`设置为绝对定位到右上角，而且是在`header[role="banner"]`的上下文中应用样式：

```
//// Banner Area Styles
//
header[role="banner"] {
  .utility-nav {
    position: absolute;
    top: 0;
    right: 0;
  }
}
```

(4) 下面我们要调整一些细节。

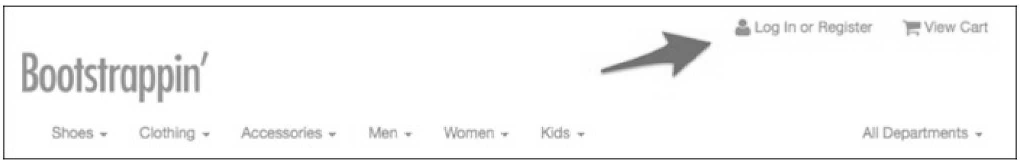
- 为`.banner-brand`类添加上内边距，以增加页头区的高度。
- 将页头区 `container` 的定位方式设置为 `relative`，以使它包含绝对定位的 `utility-nav` 元素。
- 删除无序列表的项目符号。

- ❑ 向左浮动列表项。
- ❑ 将链接显示为 inline-block 并添加内边距。
- ❑ 删除悬停时的下划线。

完成上述调整的样式规则如下：

```
header[role="banner"] {
  .banner-brand {
    padding-top: 40px;
  }
  > .container {
    position: relative;
  }
  .utility-nav {
    position: absolute;
    top: 0;
    right: 0;
    > ul {
      list-style: none;
      > li {
        float: left;
        > a {
          display: inline-block;
          padding: 8px 12px;
          &:hover {
            text-decoration: none;
          }
        }
      }
    }
  }
}
```

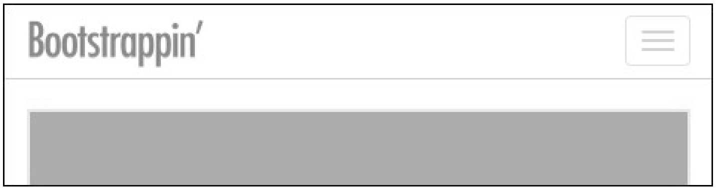
保存修改并编译。把浏览器窗口调整到桌面窗口大小，然后刷新。应该能看到 utility-nav 元素出现在了页头区的右上角位置。



这些调整适合中大型的视口。下面我们针对折叠后的响应式导航条来添加样式。

4.4　调整响应式导航

在小屏幕中，导航条折叠后 utility-nav 会出现问题。最明显的问题是它会消失不见：



要想让 utility-nav 重见天日，必须给它设置一个比导航条更大的 z-index，前者在_variables.less 中被设置为 1000。我们可以在_banner.less 中，把.utility-nav 的 z-index 设置为 1999。

```
.utility-nav {
  ...
  z-index: 1999;
```

于是，实用导航就会出现了：



接下来的问题就是它会遮挡 navbar-toggle 按钮。为此，我们要把这个切换按钮转移到导航条的左侧，步骤如下。

(1) 在编辑器中打开 less/_navbar.less。

(2) 搜索注释// Navbar toggle，编辑这条注释正下方的.navbar-toggle 选择符，把浮动值由 right 改为 left，把 margin-right 改为 margin-left：

```
.navbar-toggle {
  position: relative;
  float: left; // edited
  margin-left: @navbar-padding-horizontal; // edited
```

保存并编辑上面的修改，可以看到切换按钮已经转移到了折叠后导航条的左端，如下图所示。



嗯，不错。

现在解决过分拥挤的问题，也就是要对除屏幕阅读器之外的其他设备隐藏链接文本。在折叠后的导航条中，图标本身就足以传达意图了，何况还可以把图标弄得更大一些。

(1) 在 index.html 中，用 span 标签包围 utility-nav 中每个链接的文本：

```
<li><a href="#" title="Login or Register"><i class="icon fa
  fa-user fa-lg"></i><span>Log In or
  Register</span></a></li>
<li><a href="#" title="View Cart"><i class="icon fa
  fa-shopping-cart fa-lg"></i><span>
  View Cart</span></a></li>
```

这样可以为后面进一步调整样式提供抓手。

(2) 在_banner.less 中添加针对这些 span 标签的媒体查询。在使用 LESS 的情况下，可以精确地嵌套媒体查询。在此要使用@grid-float-breakpoint 变量，把 max-width 查询设置为@grid-float-breakpoint - 1，因为这个变量的值意味着在它那么宽时，导航条就会从折叠变成扩展状态。在这个媒体查询中，使用实用类 sr-only 作为混入，对除屏幕阅读器之外的所有设备隐藏文本。（参见这个类的文档：http://getbootstrap.com/css/#helper-classes-screen-readers。）代码片段如下：

```
.utility-nav {
  ...
  ...
  > a {
  ...
    @media (max-width: (@grid-float-breakpoint - 1)) {
      span {
      .sr-only();
      }
    }
  }
}
```

这样就隐藏了 span 标签中的文本，屏幕上将只剩图标！

(3) 再增大图标尺寸，并在垂直方向增加一些行高。同样还在这个媒体查询中写样式：

```
@media (max-width: @grid-float-breakpoint - 1) {
  span {
    .sr-only();
  }
  .icon {
    font-size: 2em;
    line-height: 1.2;
  }
}
```

保存，编译，刷新。应该看到下图所示的结果：



再放大、缩小浏览器窗口，反复经过断点宽度，看看整个页头区来回变换的效果是不是很平滑。

就我而言，能有这么一个框架让我如此高效地实现这种流畅的响应式界面，我真要谢天谢地了。

下面，我们要调整配色。

4.5 调整配色

我们网站现在的配色是标准的企业网站颜色：蓝、红、灰。下面我们把这些颜色放到变量里。

(1) 在编辑器中打开_variables.less，从一开始的变量开始。

(2) 先看一下目前灰色变量覆盖的范围。如果大家以 04_Code_BEGIN 中的文件为起点，会发现我们已经把第 2 章定义的变量继承过来了。这些变量不光在第 2 章有用，现在同样可以派上用场。

```
// Grays
// -------------------------
@gray-darker:            #222; // edited
@gray-dark:              #454545; // edited
@gray:                   #777; // edited
@gray-light:             #aeaeae; // edited
@gray-lighter:           #ccc; // edited
@gray-lightest:          #ededed; // edited
@off-white:              #fafafa; // edited
```

(3) 在灰色变量下方，再添加品牌色。修改`@brand-primary`，新增红色的`@brand-feature`：

```
@brand-primary:          #3e7dbd; // edited blue
@brand-feature:          #c60004; // added new red
```

(4) 下面调整链接的悬停颜色，使其比`@brand-primary` 稍浅（而不是稍深），否则就太深了：

```
// Links
// -------------------------
@link-color:            @brand-primary;
@link-color-hover:      lighten(@link-color, 15%);
```

设置好这些颜色变量后，就可以着手调整导航条的配色了。

4.6　调整折叠后的导航条配色

还在_variables.less 中，搜索`// Navbar`，在这里可以看到导航条用到的变量。这里指定的大多数标准值既对折叠后的响应式导航条有效，也对宽屏幕下扩展的导航条有效。

我们希望折叠后响应式导航条的背景、文本和链接颜色与默认值基本一致，但在中大型视口中变成蓝色背景、浅色文本。

为此要调整一些变量的默认值，然后再创建一些新变量，只应用给扩展后的导航条。

(1) 把`@navbar-height`的值减小为 44px，然后把前面设置的变量应用到合适的地方。再把`@navbar-default-color`设置为`@text-color`，把`@navbar-default-bg`设置为`@white`。

```
// Basics of a navbar
@navbar-height:                   44px;
...
@navbar-default-color:            @text-color;
@navbar-default-bg:               #fff;
```

(2) 向下找到导航条链接区，调整颜色让链接与导航条文本颜色一致，并给活动链接添加一点背景色：

```
// Navbar links
@navbar-default-link-color:         @navbar-default-color;
@navbar-default-link-hover-color:   @navbar-default-color;
@navbar-default-link-hover-bg:      darken(@navbar-default-bg, 5%);
@navbar-default-link-active-color:  @navbar-default-color;
@navbar-default-link-active-bg:     @navbar-default-link-hover-bg;
```

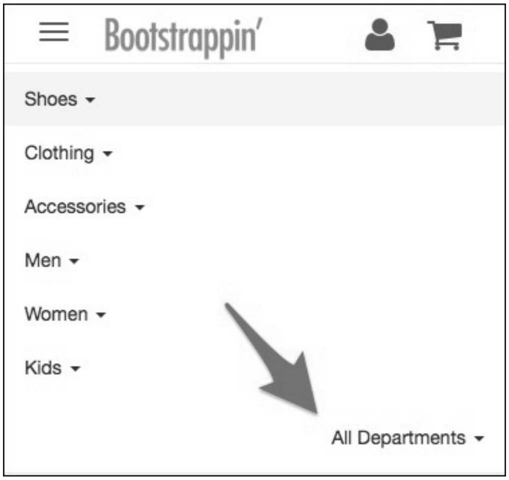
(3) 再调整`navbar-toggle`的样式，删除边框和背景，调深导航条：

```
// Navbar toggle
@navbar-default-toggle-hover-bg:           transparent;
@navbar-default-toggle-icon-bar-bg:        @gray;
@navbar-default-toggle-border-color:       transparent;
```

保存，编译，然后刷新浏览器，在窄视口中应该看到下图所示的效果：



折叠导航条还有两个地方要调整。点击打开下拉导航菜单，会看到浮动到右侧的 All Departments 链接。



还记得我们把 `pull-right` 类放到了 All Departments 菜单项中了吗？那就是为了让它浮动到扩展后的导航条右侧。但此时此地，我们希望它还在左侧，怎么办？Bootstrap 专门有一个类就是为了这个目的！

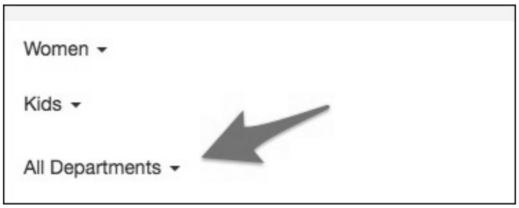
在 index.html 中找到包含 All Departments 的标记，把类 `pull-right` 改为 `navbar-right`，如下所示：

```
<ul class="nav navbar-nav navbar-right">
  <li class="dropdown">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">All
      Departments <b class="caret"></b></a>
```


想知道背后都发生了什么，就打开_navbar.less，搜索 `.navbar-right`。你会看到在一些注释下面有以下代码：

```
@media (min-width: @grid-float-breakpoint) {
  .navbar-left  { .pull-left(); }
  .navbar-right { .pull-right(); }
}
```

这些规则嵌套在一个媒体查询中，只适用于扩展后的导航条，而这些类恰好能满足我们的需要。在应用新的 `navbar-right` 类之后，保存 index.html，刷新，会发现 All Departments 在折叠的导航条中会浮动到左侧，而在扩展的导航条中仍然会浮动到右侧。



太厉害了！现在，我们来调整导航条中下拉菜单的行为。从 Bootstrap 3.0.2 起，下拉菜单项被配置为与折叠后的导航条同宽，但前提是导航条是在最初的`@screen-sm-min` 值为`@grid-float-breakpoint` 的情况下发生折叠。我们的下拉菜单就不是全宽了。要测试，可以把浏览器窗口拖动到`@screen-sm range`（768 ~ 991px），再试试打开下拉菜单。

你会看到下面的结果：



实际上，只要调整一个媒体查询，就可以解决这个问题。

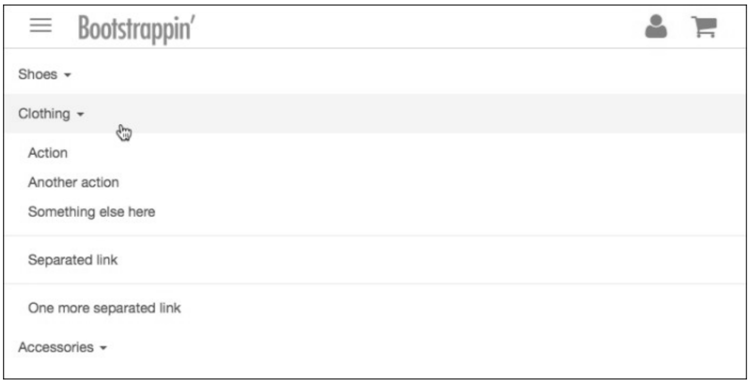
(1) 打开_navbar.less 搜索`.open.dropdown-menu`，会发现这条规则被嵌套在一个媒体查询中：

```
@media (max-width: @screen-xs-max) {
  // Dropdowns get custom display when collapsed
  .open.dropdown-menu {
```

(2) 我们要调整这个媒体查询，让它的 `max-width` 值等于`@grid-float-breakpoint`。实际上，就简单地把这个变量放到那儿就行：

```
@media (max-width: @grid-float-breakpoint) {
```

保存修改，编译文件，然后刷新页面。这次就会看到下拉菜单与屏幕一样宽了：



漂亮。接下来轮到水平导航条了。

调整水平导航条

在中大型屏幕中，导航条水平排列在 Logo 下面。我们希望此时的导航条呈现`@brand-primary` 变量中设置的蓝色背景。为此，必须要反转链接和文本的颜色，即由深变浅。我们要使用 Bootstrap 的 `inverted-navbar` 变量和样式。

(1) 在_variables.less 中，搜索注释`// Inverted navbar`。找到后，会发现一些与默认导航条所用类似的变量。我们就要通过它们来给扩展后的导航条应用颜色。

(2) 按照如下所示调整变量：

```
// Inverted navbar
//
// Reset inverted navbar basics
@navbar-inverse-color:              @gray-lightest;
@navbar-inverse-bg:                 @brand-primary;
@navbar-inverse-border:             darken(@navbar-inverse-bg, 10%);

// Inverted navbar links
@navbar-inverse-link-color:         @navbar-inverse-color;
@navbar-inverse-link-hover-color:  #fff;
@navbar-inverse-link-hover-bg:     darken(@navbar-inverse-bg, 5%);
@navbar-inverse-link-active-color:@navbar-inverse-link-hover-color;
@navbar-inverse-link-active-bg:    darken(@navbar-inverse-bg, 10%);
```

调整好这些变量后，只要把它们应用给扩展导航即可。为此得写几行自定义的 LESS 代码。考虑到这种颜色切换属于页头匹配色的变化，所以我们就把代码写到_banner.less 中。

(3) 打开_banner.less 并添加一个新的带注释的区块：

```
// Apply .navbar-inverse styles to the expanded navbar
@media (min-width: @grid-float-breakpoint) {
  .navbar-default {
    .navbar-inverse();
  }
}
```

这个媒体查询使用`@grid-float- breakpoint` 变量确定了应用新规则的最小视口宽度。因为我们已经在导航条中添加了 `navbar-default` 类，所以可以直接使用这个类作为选择符。混入`.navbar-inverse()`则把在_navbar.less 中定义的样式`.navbar- inverse`应用给了这个媒体查询中的导航条。

保存以上修改，编译文件，刷新浏览器。在中大型屏幕中，可以看到导航条的蓝色背景和浅色文本了。



此时此刻，可以看到导航条两端的圆角。我们得把这些样式去掉。为此，打开_variables.less，搜索变量@navbar-border-radius，然后将其值设置为 0：

```
@navbar-border-radius:              0;
```

最后，我们把文本转换为大写形式，稍微缩小一点，再加粗。

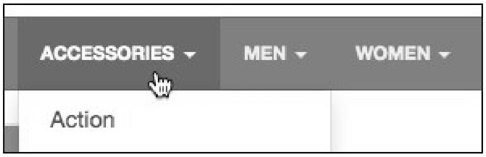
在_banner.less 中，把如下代码添加到.navbar-inverse()混入之后：

```
@media (min-width: @grid-float-breakpoint) {
  .navbar-default {
    .navbar-inverse();
    .navbar-nav > li > a {
      text-transform: uppercase;
      font-size: 82%;
      font-weight: bold;
    }
  }
}
```

这样就得到了我们想要的结果：



下图是悬停在一个导航项上的特写：

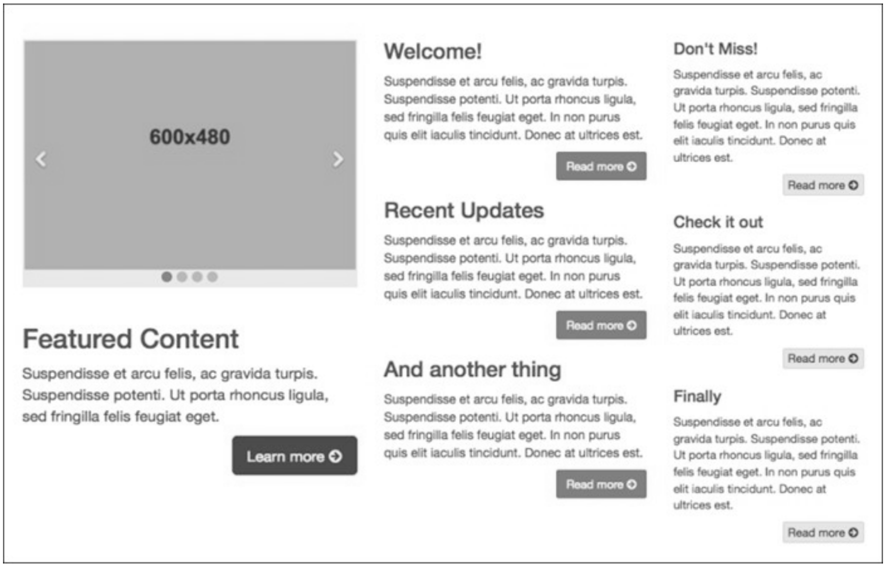


我们的页头和导航条完工了！接下来该轮到页面的主内容区了。

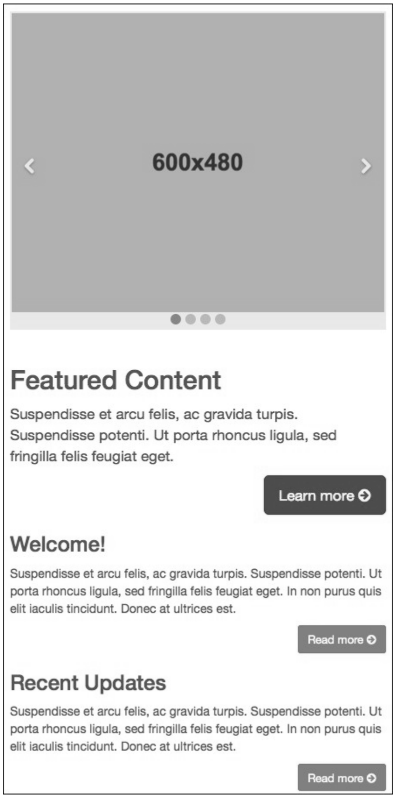
4.7　设计复杂的响应式布局

假设我们在刚刚结束的客户会面中作出了一个承诺，要把主页内容分成三层，按重要程度排序。

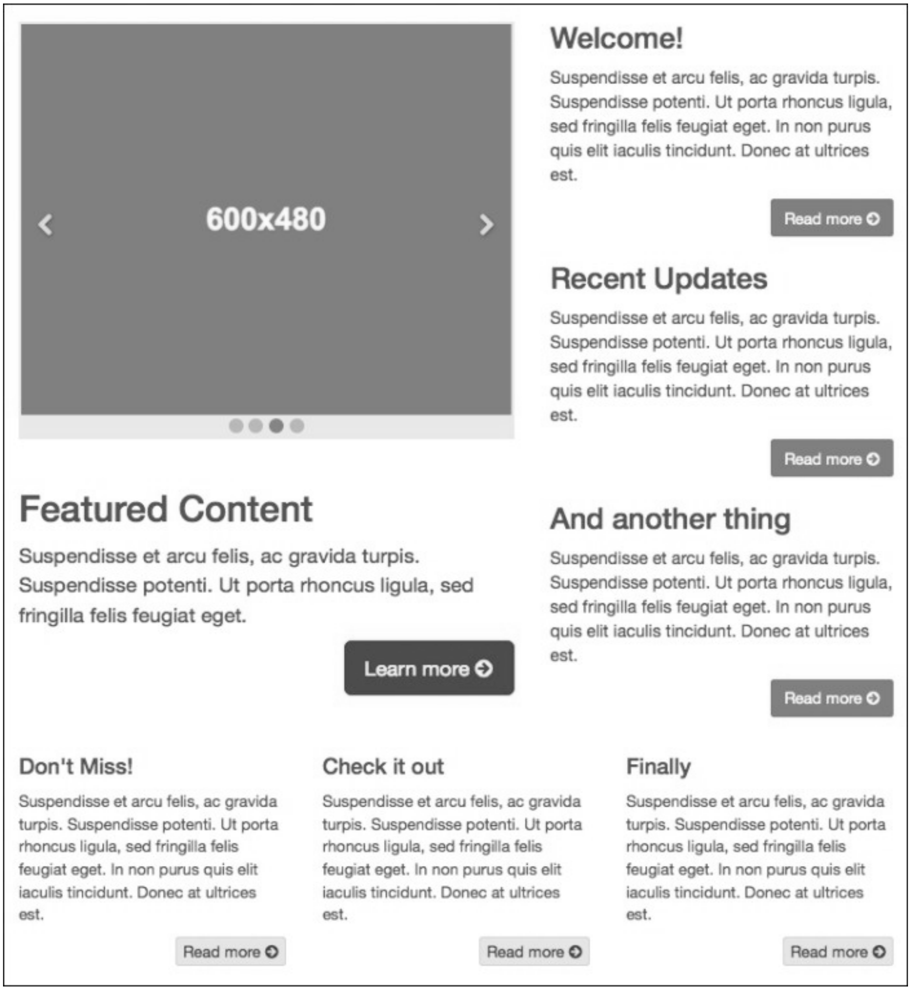
在中大型视口中，所有内容将分布在三栏中，如下图所示：



在较窄的视口中，这些栏将从上到下排成一栏：



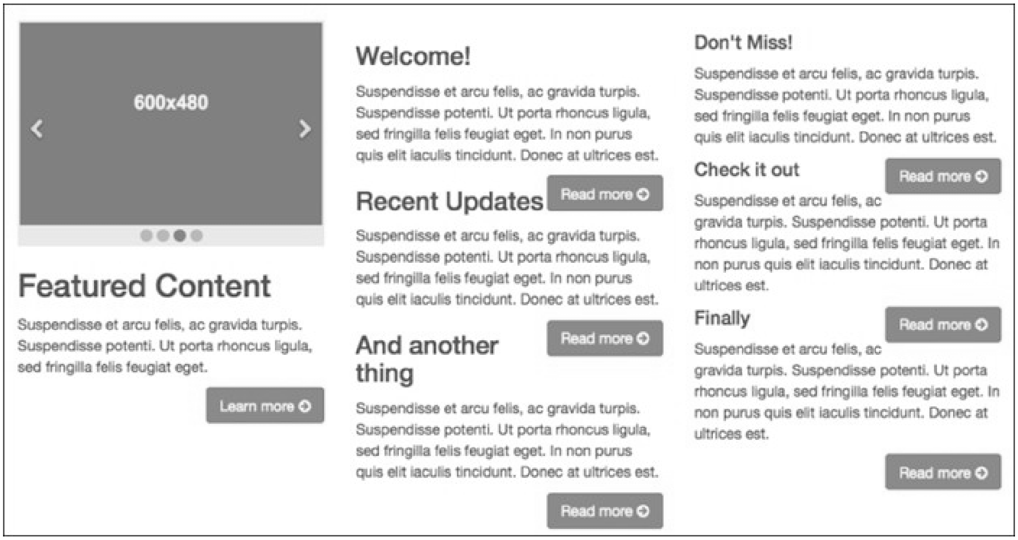
而在平板电脑的视口中，并排的只有两栏，第三栏水平放到它们下面，如下图所示：



作为起点，我们已经有了三个等宽栏的标记。下面就来看一下这些标记，考虑一下怎么实现我们的设计意图。先从中大型屏幕的三栏布局开始。

4.7.1　调整中、宽布局

当前，在中宽视口中，三栏是等宽的，而且字体大小、按钮大小，还有颜色都一样。结果就是没有层次感，如下图所示：



要实现内容从视觉上的分层，可以调整栏宽、字体大小、按钮大小，还有颜色。我们先从调整栏宽开始。

(1) 在 index.html 中，搜索包含内容的 section 标签：

```
<section class="content-primary col-sm-4">
```

这里的类 col-sm-4 表示当前栏是父元素宽度的三分之一，从小视口（764px）及以上宽度开始。

我们想在中大视口（992px 及以上）内保留三栏，而且希望第一栏比另两栏宽。

(2) 把 col-sm-4 修改为 col-md-5，如下所示：

```
<section class="content-primary col-md-5">
```

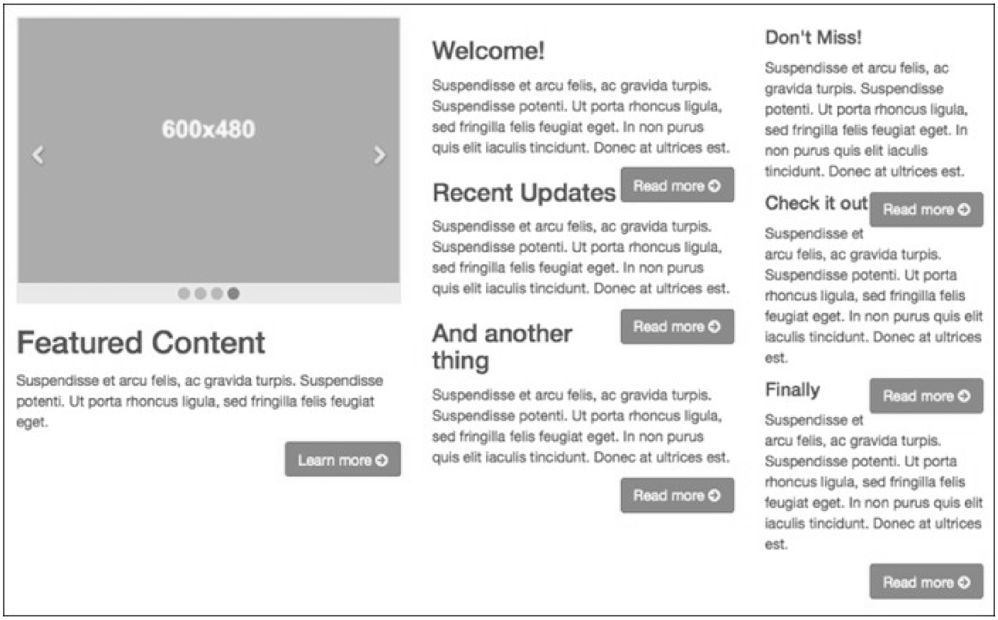
这样就把栏宽设置为了父元素的 5/12，并且从中型视口开始应用。

(3) 再搜索到后面两栏的开始 section 标签，将它们的类分别改为 col-md-4 和 col-md-3：

```
<section class="content-secondary col-md-4">
...
<section class="content-tertiary col-md-3">
```

保存，刷新，可以看到以宽度分层的三栏：

4



中间和第三栏的按钮并没有清除。下一步就来调整这些按钮，还有字体大小。

4.7.2　调整标题、字体大小和按钮

我们先来调整标题，以便它们清除自己上方的按钮，目前这些按钮都浮动到了右侧。为此，要用到之前新建的用于管理页面内容细节的文件：_page-contents.less。

以下是调整步骤。

(1) 在_page-contents.less 中，写一个选择符选择嵌套在 Bootstrap 的分栏类中的 `h1` 到 `h4`。这里可以使用 CSS2 的属性选择符，同时只针对嵌套在类以 `col-`开头的元素内的这些标题：

> 本章后面还要设计包含自己的一组分栏的页脚。因此，这里还要确保把规则嵌套在针对 `main` 元素的选择符内。

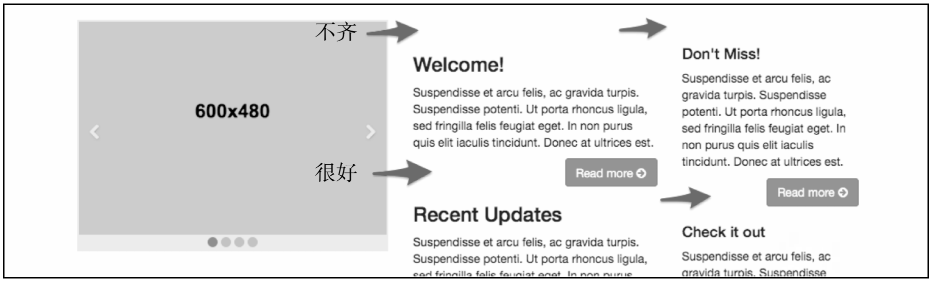
这样，就可以选中所有可能用到的标题标签，以便清除它们的浮动，再给它们添加一些内边距。

```
main {
...
  [class*="col-"] {
    h1, h2, h3, h4 {
      clear: both;
```

```
      padding-top: 20px;
    }
  }
}
```

这样标题之间有了必要的分隔，也让按钮浮动到了相应位置。但这样也在第二和第三栏上方增加了不必要的上内边距。

在下面的屏幕截图中，下方箭头指出的是改进，上方箭头指出的是内边距造成的两栏顶部不齐的问题。



(2) 下面来删除每栏最顶部标题的上内边距和上外边距。为此，要使用`:first-child`选择符，嵌套在标题选择符内。这里使用的&组合符，用于选择这些标题的第一个实例：

```
h1, h2, h3, h4 {
  ...
  &:first-child {
    margin-top: 0;
    padding-top: 0;
  }
}
```

(3) 结果就是去掉了后两栏顶部多余的内外边距：



(4) 不过，我们只想在小或较大视口中删除上内、外边距，这时候主页呈现为多栏。显然，应该把上面的样式嵌套在与相应断点对应的媒体查询中。这个断点就是从单栏布局切换到多栏布局的断点。

换句话说，需要把上面的样式嵌套在小及更大的视口中：

4

```
@media (min-width: @screen-sm-min) {
  &:first-child {
    margin-top: 0;
    padding-top: 0;
  }
}
```

通过把新样式嵌套在上面的媒体查询中，可以保留单栏布局下元素间适当的间距，如下图所示：



完成了上述调整，接下来可以调整按钮和字体大小，从而反映内容层次了。下面就来增大主内容区字体和按钮大小，还有修改颜色。

4.7.3　增大主栏

首先来增大主栏内容的字体大小。

(1) 在_variables.less 中，搜索`@font-size-large` 变量，将其值修改为：

```
ceil(@font-size-base * 1.15);
```

(2) 在 in _page-contents.less 中，添加如下代码，以利用上一步中设定的字体大小：

```
.content-primary {
  font-size: @font-size-large;
}
```

保存修改，编译文件，刷新浏览器。应该看到主栏文本的字体增大了！

接下来调整按钮的颜色，这要用到红色的`@brand-feature` 变量，这个变量是在_variables.less 中设定的：

```
@brand-feature:        #c60004;
```

还需要利用 Bootstrap 在 mixins.less 中提供的方便的混入。希望大家抽点时间看看这个文件。打开 bootstrap/mixins.less，搜索`// Button`，可以看到以下面代码开头的混入：

```
.button-variant(@color; @background; @border) {
```

这个混入的作用如下。

- 指定按钮字体、背景和边框颜色（分别对应于混入接受的三个参数）。
- 生成悬停、焦点和禁用状态的按钮，调整字体颜色、背景颜色和边框。

如果你有兴趣，还可以看一看 Bootstrap 在 bootstrap/buttons.less 中如何使用这个混入，就在注释`// Alternate buttons`的下面。以下是为默认主按钮生成的样式：

```
// Alternate buttons
// --------------------------------------------------
.btn-default {
  .button-variant(@btn-default-color; @btn-default-bg; @btn-default-border);
}
.btn-primary {
  .button-variant(@btn-primary-color; @btn-primary-bg; @btn-primary-border);
}
```

> 以@btn-default-和@btn-primary-开头的变量是在 variables.less 中定义的。

按照同样的模式，只需简单四步即可生成我们自己的自定义功能按钮。

(1) 首先，准备一组新的按钮变量。在 In _variables.less 中，`// Buttons`下面，复制三个`@btn-primary-`变量，将`-primary-`改为`-feature-`，并使用`@brand-feature`作为背景颜色：

```
@btn-feature-color:              #fff;
@btn-feature-bg:                 @brand-feature;
@btn-feature-border:             darken
  (@btn-feature-bg, 5%);
```

(2) 然后，创建一个文件来保存自定义按钮的样式。新建_buttons-custom.less 并根据 bootstrap/buttons.less 中的混入写一个下面这样的混入调用：

```
.btn-feature {
    .button-variant(@btn-feature-color; @btn-feature-bg; @btn-feature-border);
}
```

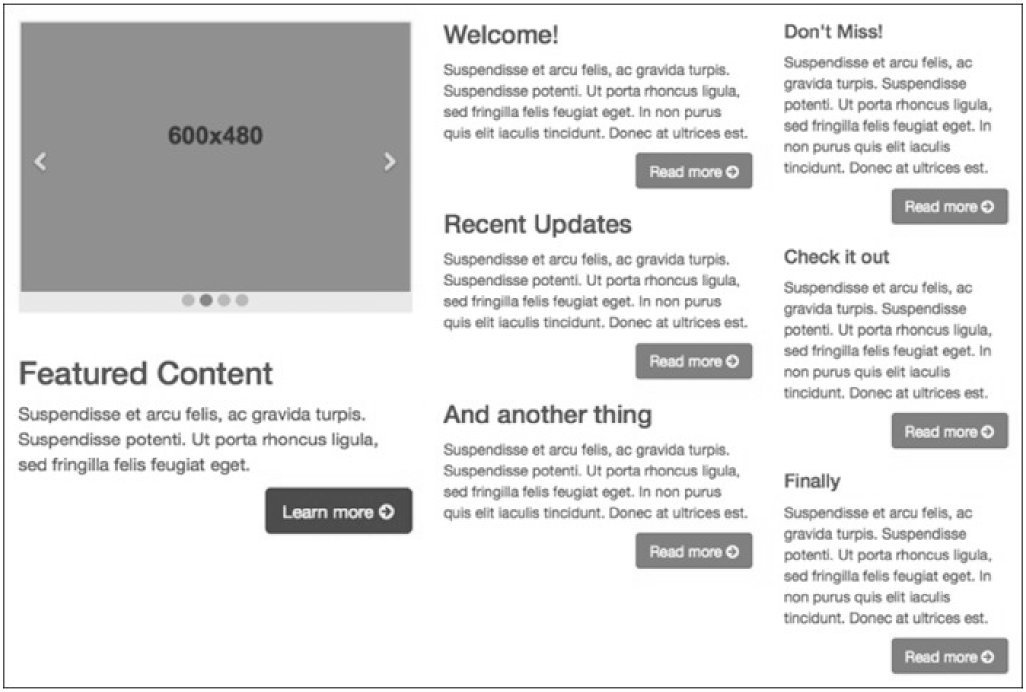
(3) 保存文件，并将其添加到__main.less 中的导入列表：

```
@import "bootstrap/buttons.less";
@import "_buttons-custom.less"; // added
```

(4) 在 index.html 中，将类名`btn-primary`改为`btn-feature`。完事之后，还要把按钮变大一些，因此再添加类`btn-lg`：

```
<a class="btn btn-feature btn-lg pull-right" href="#">
  Learn more
```

保存并刷新浏览器，应该看到如下结果。左侧主栏的字体大小和按钮都变大了，而且使用了 brand-feature 颜色。



与此同时，第二（中）栏的字体大小和按钮颜色正是我们想要的。接下来需要修改的是降低第三栏内容的重要程度。

4.7.4　调整第三栏

对第三栏要做的很简单，就是缩小字体大小，同时让按钮不那么突出。

(1) 首先调整字体大小。在_variables.less 中，调整@font-size-small 变量：

```
@font-size-small:          ceil(@font-size-base * 0.90);
```

(2) 接下来把下面这行添加到_page-contents.less 中：

```
.content-tertiary {
  font-size: @font-size-small;
}
```

(3) 保存，编译，刷新，字号变小了。

(4) 接下在，要编辑 index.html 中的按钮类，把 btn-primary 改为 btn-default，并使用 btn-xs 减小其尺寸：

```
<a class="btn btn-default btn-xs pull-right" href="#">Read more ...
```

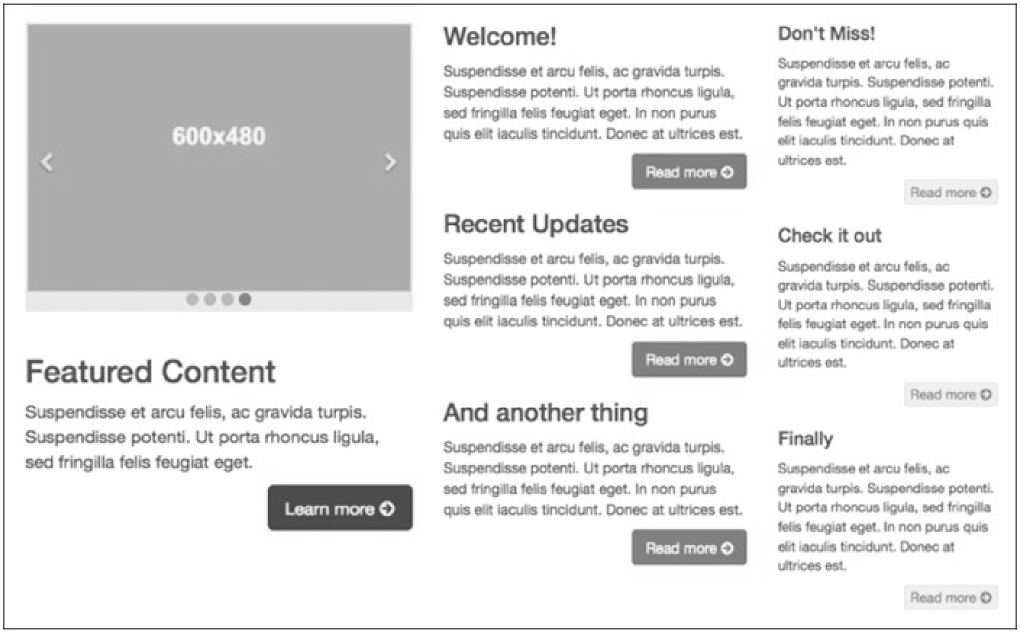
这样就减小了按钮并把背景变成了白色。

(5) 再把按钮的背景颜色改为浅灰色，同时调整字体颜色和边框。在_variables.less 中，像下面这样调整三个@btn-default-变量的值：

```
@btn-default-color:              @gray;
@btn-default-bg:                 @gray-lightest;
@btn-default-border:             darken
  (@btn-default-bg, 5%);
```

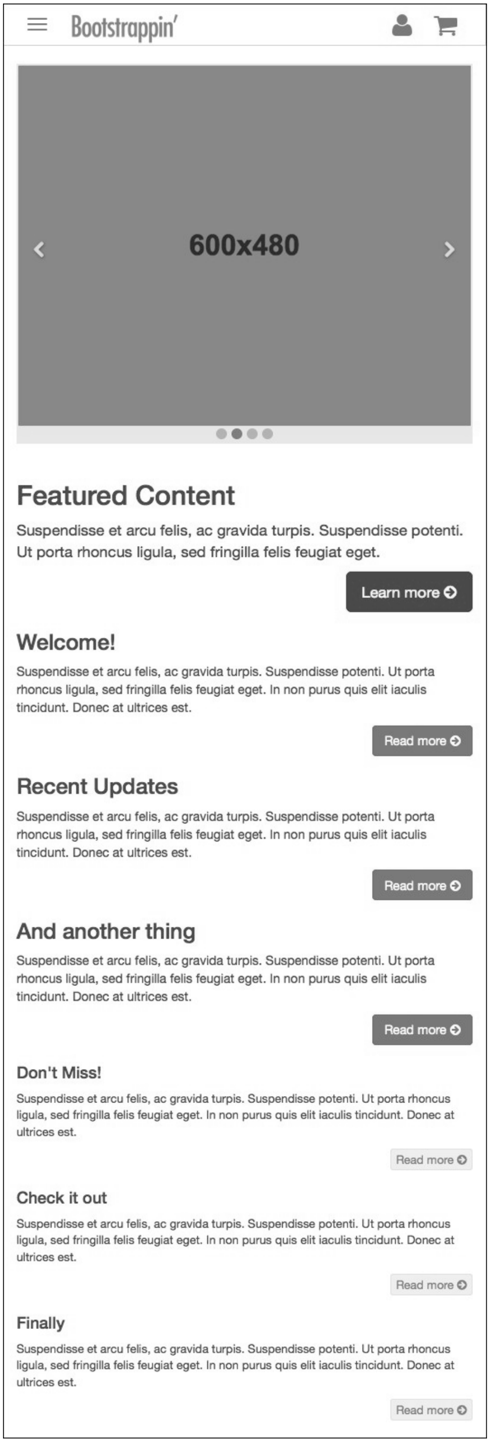
保存修改，编译文件，刷新浏览器。

现在页面的层次已经很清晰了，从左到右依次是主内容、次内容和第三栏。



再看看我们的设计在小屏幕单栏布局时的样子：

4


Bootstrappin'
600x480
Featured Content
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget.
Learn more
Welcome!
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more
Recent Updates
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more
And another thing
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more
Don't Miss!
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more
Check it out
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more
Finally
Suspendisse et arcu felis, ac gravida turpis. Suspendisse potenti. Ut porta rhoncus ligula, sed fringilla felis feugiat eget. In non purus quis elit iaculis tincidunt. Donec at ultrices est.
Read more

在窄视口中，三栏布局变成了垂直排列的一栏，主内容在上，然后是次内容和第三栏。

那么剩下所要做的，就是对设计精雕细琢，以便让它在不同设备和视口中体验更佳。

4.7.5　针对多个视口进行微调

无论在什么视口中，通常都应该在页面中提供一些留白。另外，每个区块的边框最好也有所标示。

(1) 首先，在内容上下各添加一些内边距。给 main 元素添加一些上内边距，这个内边距适用于所有视口，所以不必使用媒体查询：

```
main {
  padding-top: 20px;
  padding-bottom: 40px;
}
```

(2) 然后，设置分栏在单栏布局时清除上方的浮动元素。如果不设置，第二和第三栏可能会覆盖紧上方的按钮。这些样式要写在媒体查询中，以便限制它只应用到窄视口：

```
//Make columns clear floats in narrow viewport single-column layout
@media (max-width: @screen-sm-min) {
  [class*="col-"] {
    clear: both;
  }
}
```

就这样了，主内容区收工。最后是复杂的页脚。

4

4.8　复杂的页脚

接下来我们要实现一个复杂的多用途的页脚，页脚包括：指向网站三个重要栏目的三组链接、About Us 文本、社交媒体图标，还有 Logo。

4.8.1　准备标记

我们先从准备标记着手。页脚的目的是对用户尽可能有用，其标记可以按如下步骤来准备。

(1) 用编辑器打开项目文件夹 04_Code_BEGIN 中的 footer-content.html，复制其中的全部内容。

(2) 再回到 index.html，找到粘贴它的位置。这个位置就在 footer role="contentinfo" 中，位于 div class="container"后面，ul class="social"前面。（我已经在页脚的位置放了注释。）

(3) 粘贴内容之前，我们再准备一下利用 Bootstrap 的网格系统。为此，像下面这样把页

脚区包含在 div class="row"中：

```
<footer role="contentinfo">
  <div class="container">
    <div class="row">
      <!-- INSERT ADDITIONAL FOOTER CONTENT HERE -->
    </div><!-- /.row -->
    <ul class="social">
```

(4) 下面把内容粘贴到位。

(5) 接下来把三组链接及各自的标题包含在类为 col-md-2 的 div 中。这样就保证在中大视口中每一组链接的宽度是父元素总宽度的六分之一。加在一块，三组链接占视口宽度的一半。

```
<div class="col-md-2">
  <h3>Categories</h3>
```

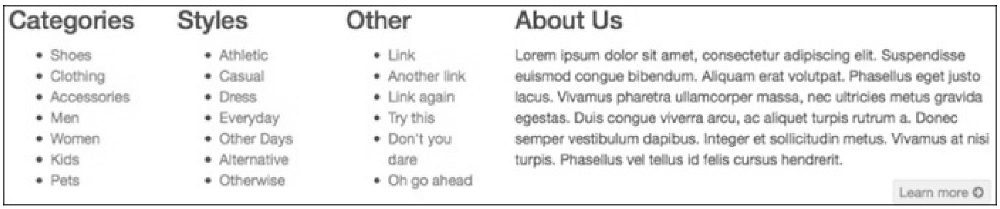
(6) 继续把这一行内容做完，把 About Us 标题及其段落包含在一个类为 col-md-6 的 div 中，这样它就占据了剩下的一半宽度。

```
<div class="about col-md-6">
  <h3>About Us</h3>
```

> 别忘了给每个 div 补全结束标签。

(7) 保存，刷新，检查结果。

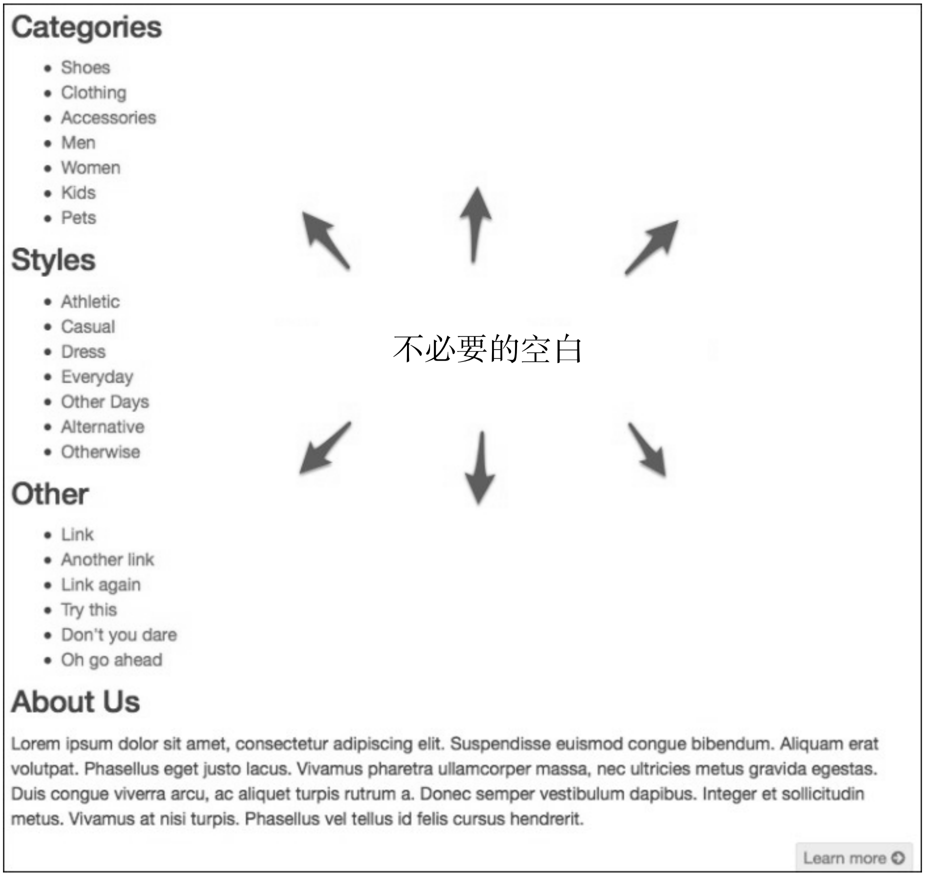
在 980px 及更大的视口中，页脚中的栏如下所示：



这是中大型视口中的布局。在超小的屏幕中，呈现出的则是单栏布局。但对于 768px 到 980px 之间的平板电脑，我们的布局还是需要调整一下。开始吧。

4.8.2　调整布局适应平板

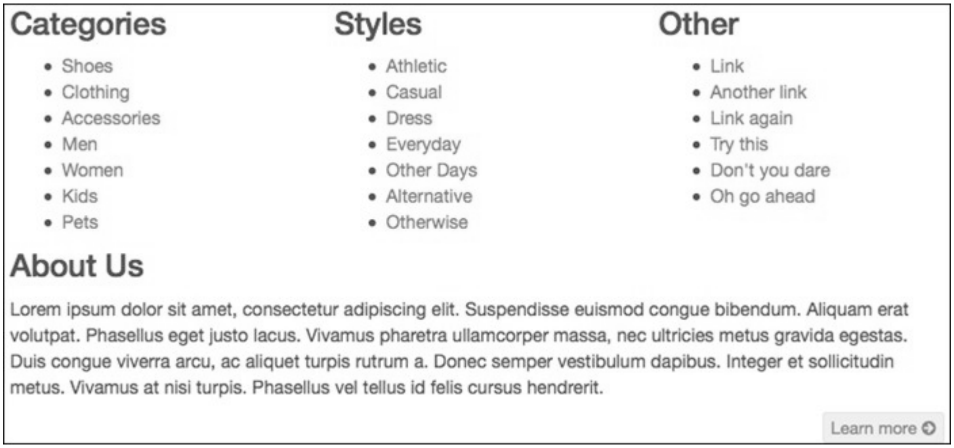
测试一下视口在 768px 到 980px 之间时的布局。Bootstrap 把这个区间界定为小断点，对应变量@screen-sm 和 col-sm-网格类。在这个宽度内，单栏布局会导致不必要的空白，如下图所示：



要改进这个布局，可以让三组链接浮动起来。使用 Bootstrap 的 `col-sm-`类 `col-sm-4`，可以将每一栏设置为三分之一宽，使用 `ol-sm-12` 将 About Us 设置为全宽：

```
<div class="col-sm-4 col-md-2">
...
<div class="col-sm-4 col-md-2">
...
<div class="col-sm-4 col-md-2">
...
<div class="about col-sm-12 col-md-6">
```

保存并在小视口中测试一下，应该能看到下图所示的结果：



改进很多啊！不过离完成还远着呢。尝试单击上方三栏中的链接，恐怕点击不了。检查元素会发现包含 About Us 栏的 `div` 元素并没有清除上方的浮动栏。虽然 About Us 标题及文本会出现在三个浮动栏下方，但那个 `div` 元素却仍然覆盖在三栏内容上面。

4.8.3　针对性地清除

在 Bootstrap 标准的布局方案中，应该使用类为 `row` 的 `div` 元素清除上方的浮动栏。但我们在此要使用另一个方案，因为我们只希望这个内容块在特定的断点范围清除浮动。

为此，可以在 LESS 文件中写一些自定义的样式。不过，也可以直接在标记中使用 Bootstrap 的响应式实用类提供的针对性的 `clearfix`。因为我们已经在标记中指定了网格类，那就干脆使用第二个方案。

关于我们使用的这个方案，大家可以参考 Bootstrap 的文档，地址在：http://getbootstrap.com/css/#grid-responsive-resets。如此一来，就要创建一个类为 `clearfix` 的 `div`，并添加一个 Bootstrap 的响应式实用类，使其只在小屏幕中可见。就把这个 `div` 放到 About Us 栏的前面吧：

```
<div class="clearfix visible-sm"></div>
<div class="about col-sm-12 col-md-6">
```

这个 `clearfix` 类会强制当前元素清除上方的浮动。而 `visible-sm` 类则控制这个 `div` 仅在小屏幕，也就是我们指定的断点范围内可见。在其他断点区间，这个 `div` 元素就像不存在一样。

保存以上修改，刷新浏览器。这次就会看到 About Us 栏清除了它上方的浮动，而链接也可以点击了。

大功告成。最后再稍微修整几处。

4.8.4 修整细节

对于页脚，我们还想再修整几个地方，包括：

- ❑ 修整三组链接的外观；
- ❑ 调整内外边距；
- ❑ 反转配色方案，与导航条保持一致。

要完成以上工作，得写一些自定义的样式。我们遵照层叠原理，先写一些针对页脚的通用规则，然后再过渡到特殊规则。

(1) 在编辑器中打开_footer.less 以添加针对页脚的自定义样式。

在这个文件里，可以看到第 2 章中一些样式的修改版本。包括给页脚添加的内边距，以及针对社交媒体图标，还有页脚中 Logo 的样式规则。

(2) 现在开始添加针对复杂页脚的样式。首先，缩小页脚字体大小，反转配色与导航条对应——蓝色背景，浅色文本。我们先设置成这样的颜色，然后再把它们稍微调暗一点。为此，就要使用_variables.less 中适当的变量，包括 @font-size-small、@navbar-inverse-bg 和@navbar-inverse-color：

```
footer[role="contentinfo"] {
  padding-top: 24px;
  padding-bottom: 24px;
  font-size: @font-size-small;
  background-color: darken(@navbar-inverse-bg, 18%);
  color: darken(@navbar-inverse-color, 18%);
```


> 这里及后面的规则，都要放到 footer[role="contentinfo"]选择符下。

(3) 接下来调整链接和按钮，以适应新的配色。同样要把规则放在 footer[role="contentinfo"]选择符下：

```
a {
  color: @navbar-inverse-color;
  &:focus,
  &:hover,
  &:active {
    color: @navbar-inverse-link-hover-color;
  }
}
.btn-default {
  color: darken(@navbar-inverse-bg, 18%) !important;
}
```

(4) 下面是四个 h3 标题，调整它们的字号，去掉下外边距，并把文本转换成大写：

```
h3 {
  font-size: 120%;
  margin-bottom: 4px;
  text-transform: uppercase;
}
```

(5) 然后，再去掉链接列表前的项目符号，调整它们的内外边距：

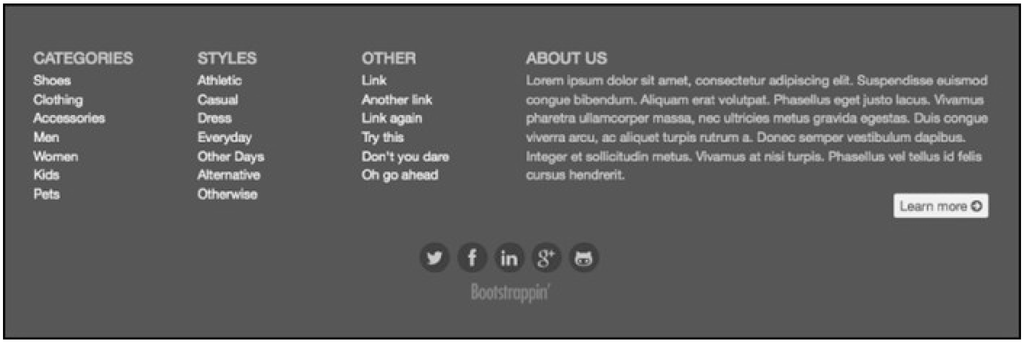
```
ul {
  list-style: none;
  padding: 0;
  margin: 0;
}
```

> 在这个设计中，这里规则针对的是站点页脚中的所有无序列表。如果你自己网站中包含其他无序列表，可能要考虑添加特定的类，比如 footer-nav。

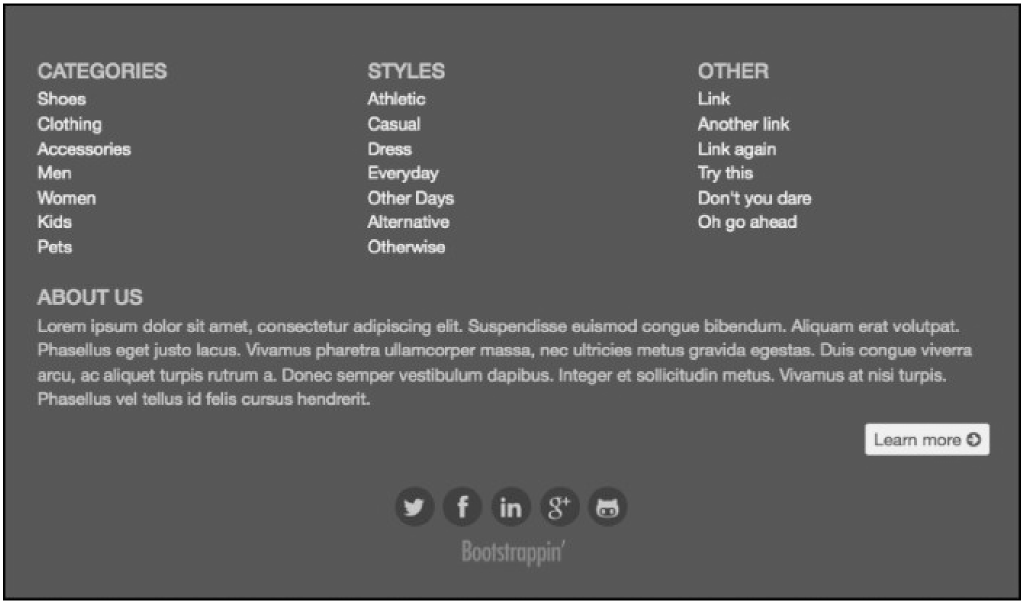
(6) 最后，调整社交媒体图标。就是添加一些上内边距，调整一下颜色，以便与新配色方案协调一致。因为图标使用的是 Font Awesome 字体，所以只要调整颜色和背景颜色的值即可：

```
ul.social {
  ...
  padding: 24px 0 0;
  ...
  > li {
    ...
    > a {
      ...
      background-color: darken(@navbar-inverse-bg, 27%);
      color: darken(@navbar-inverse-color, 18%);
      ...
      &:hover {
        ...
        background-color: darken(@navbar-inverse-bg, 32%);
        color: @navbar-inverse-link-hover-color;
      }
    }
  }
}
```

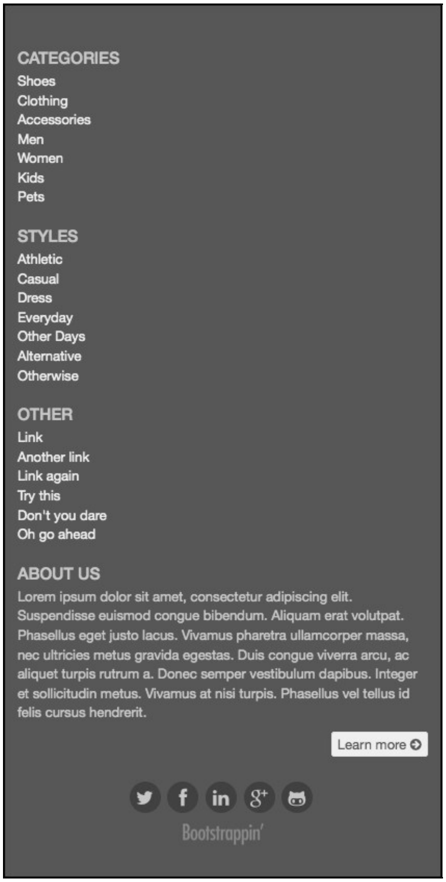
就这样了。保存，编译，刷新，好好欣赏吧！以下是页脚在中大型屏幕中的结果：



在小屏幕中的效果如下：



在超小屏幕中的效果如下：



不错嘛！我们的页脚其实挺复杂的，内容多，又适配了超小、小、中、大型视口。

4.9　小结

通过本章的项目，我们又掌握了一些利用 Bootstrap 的新技术。可以简单总结如下。

- 为复杂的响应式导航条添加样式，使其在中大视口中出现在 Logo 下方，而在小屏幕中又能折叠起来。
- 构建了自定义的响应式实用导航条，文本和图标都能创造性地适应较大和较小的屏幕。
- 为页面的主内容设计了响应式布局，使三栏内容主次分明。
- 构建了一个复杂的页脚，有效地组织了多个链接块，还有跨视口的文本段落。
- 以导航条配色为基础增强了页脚的配色。

恭喜！下一章，我们将以上述技术为依托，为这个网站的电子商务模块设计一个产品页面。

5 电子商务网站

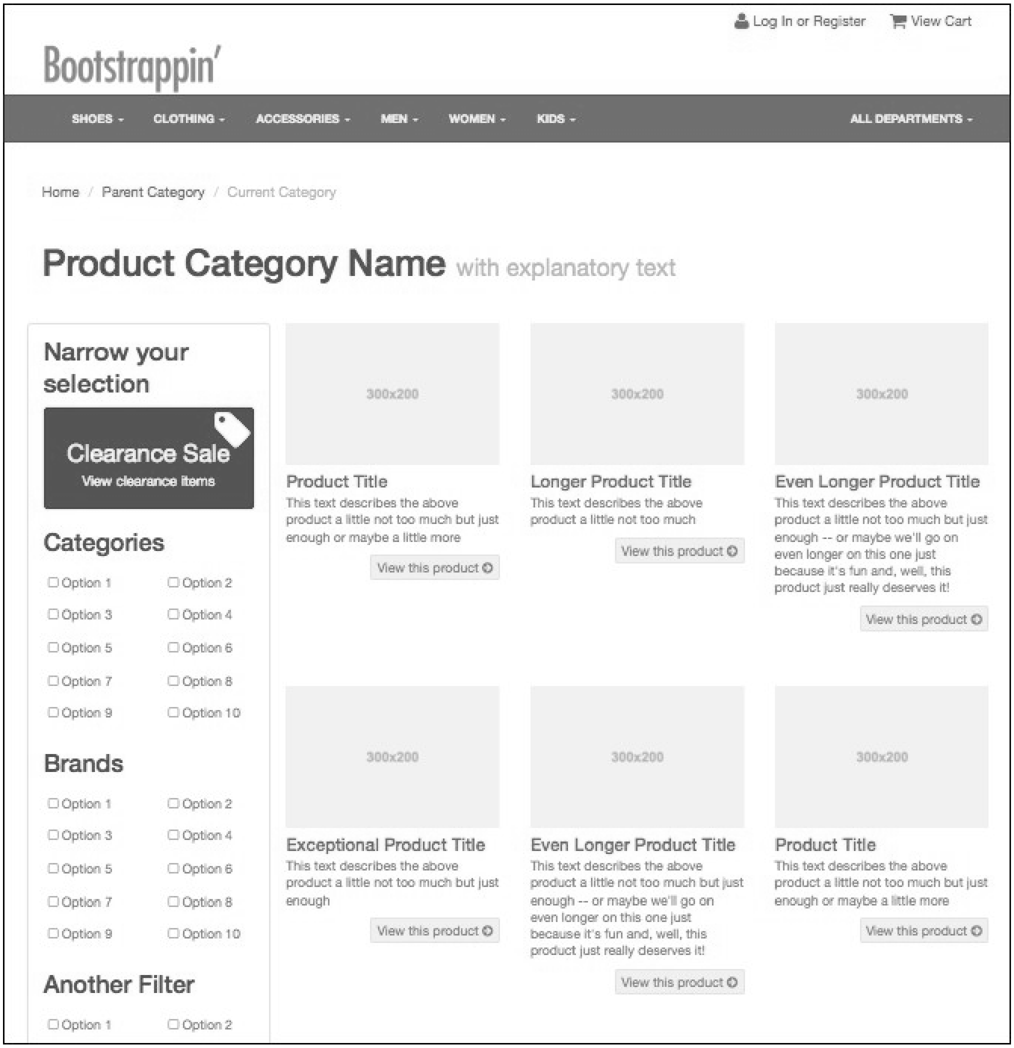
构建了公司网站之后，接下来就可考虑设计一个在线商店了。

本章的设计以上一章的设计为基础，只是添加了一个包含如下元素的新页面：

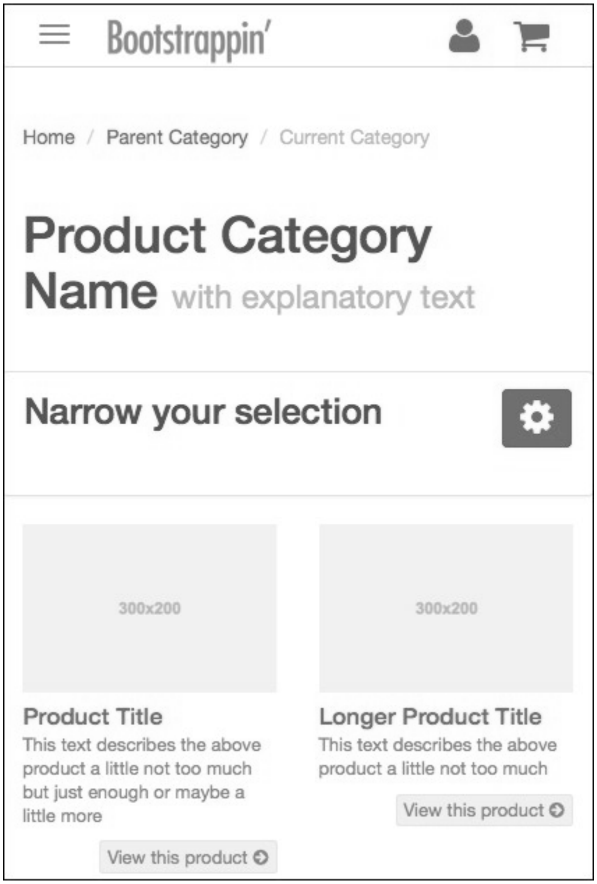
- ❑ 包含商品小图、标题和说明的产品网格；
- ❑ 位于左侧的边栏，用于按类别、品牌等筛选商品；
- ❑ 方便用户导航的面包屑和分页链接。

大家可以先看一看 Zappos（http://www.zappos.com）和 Amazon（http://www.amazon.com）的网站，搜索或者浏览一下其中的商品。本章所要创建的页面，就包含与之类似的商品网格。

完成后的设计在大、中、小屏幕中的效果应该如下图所示：



在超小屏幕上，我们希望页面的布局变成如下所示：



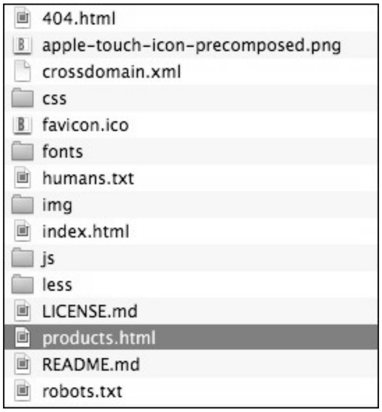
Bootstrap 为完成本章的设计提供了很好起点，在此基础上，我们要使用 LESS 完成调整工作。

5.1 商品页的标记

本章的练习文件可以在文件夹 05_Code_BEGIN 中找到。这个项目直接以第 4 章的企业网站为基础，如果有项目文件让你不理解，请参考第 4 章。

> 本书练习文件可以在这里下载：http://packtpub.com/support。

对本章而言，只有一个文件是新的，那就是 products.html。



在编辑器中打开 products.html，看一下其中的标记。

标记中的 `head`、页头、导航条元素都和第 4 章一样。不一样的地方在 `main role="main"` 元素中，这个元素中包含的新内容按先后顺序是：

- 用无序列表生成的面包屑导航链接；
- 用 `h1` 表示的页面标题；
- 一系列用于筛选商品的选项；
- 九种商品，分别带有小图、标题、说明和按钮；
- 用无序列表生成的分页链接，位于商品之下，站点页脚之上。

如果你在浏览器中打开这个文件，会发现很多地方都没有完成。面包屑看起来还不像面包屑，筛选选项还是一堆无序列表，商品展示区也不齐整（有的商品甚至错位）。

面对这个乱糟糟的局面是不是有点手足无措了？不要紧，下面我们马上就来做点什么。

- 首先要应用 Bootstrap 内置的面包屑、页面标题和分页链接的样式，再对它们进行自定义。
- 然后，我们要改进九个商品的布局，创造性地使用 Bootstrap 的网格系统，让这些格子在不同断点切换时保持视觉上的整齐划一。
- 最后，就是要给筛选选项添加样式，主要是增强它们的布局，再使用 Font Awesome 图标作为复选框。

有了规划，下面就动手吧。

5.2　面包屑、页面标题和分页导航

接下来，我们要把 Bootstrap 的样式应用到面包屑、页面标题和分页导航，然后再对它们进行自定义，以适应我们的设计。

(1) 在编辑器中打开 products.html。

(2) 找到位于页面标题 h1 上方的无序列表，给 ul 标签添加类 breadcrumb，然后给最后一个列表项添加类 active：

```
<ul class="breadcrumb">
  <li><a href="#">Home</a></li>
  <li><a href="#">Parent Category</a></li>
  <li class="active">Current Category</li>
</ul>
```

这两个类对应相关 Bootstrap 的面包屑样式，相关文档请参考这里：http://getbootstrap.com/components/#breadcrumbs。

保存并刷新浏览器。应该能看到如下图所示的效果：



(3) 接下来自定义面包屑的设计，去掉浅灰色背景和多余的内边距。对于这个简单的调整，我们可以直接修改 boostrap 文件夹中的 breadcrumbs.less 文件， 同时注释掉不需要的行，这样可以留下修改的痕迹。

把 padding 设置为 0，完全删除 background-color，使用注释可以让我们以后知道在这里修改了什么：

```
.breadcrumb {
  padding: 0; // 8px 15px; // edited
  margin-bottom: @line-height-computed;
  list-style: none;
  // background-color: @breadcrumb-bg; // edited
```

(4) 下面是页面标题。Bootstrap 的页面标题需要被嵌套在类为 page-header 的 div 元素中。相关文档可以参考这里：http://getbootstrap.com/components/#page-header。

我们按照文档来调整标记，为了利用 Bootstrap 给标题注释添加的样式，再在 small 标签中添加一些文本内容：

```
<div class="page-header">
  <h1>Product Category Name <small>with explanatory
    text</small></h1>
</div>
```

这样就会得到如下所示的结果：



(5) 保留标题带的内、外边距，删除下方边框。打开 bootstrap 文件夹中的 type.less 文件，搜索.page-header 并把规则 border-bottom 注释掉：

```
.page-header {

  // border-bottom: 1px solid @page-header-border-color;
}
```

保存，刷新，就会看到更清爽的结果。更多的空白与我们整体设计保持了一致，如下图所示：



(6) 最后是分页链接。相关的标题就在关闭的 main 标签（</main>）稍微靠上一点。在这个标题之上，依次是.container、.row 和.products-grid 的关闭 div 标签：

```
      </div><!-- /.products-grid -->
    </div><!-- /.row -->
  </div><!-- /.container -->
</main>
```

> Bootstrap 中分页样式的文档在这里：http://getbootstrap.com/components/#pagination。

要应用分页导航样式，只需把 class="pagination"添加到关闭的.products-grid 标签之上的 ul 标签：

```
<ul class="pagination">
  <li><a href="#"><span class="fa fa-chevron-left"></span>
   Prev</a></li>
  <li><a href="#">1</a></li>
  <li><a href="#">2</a></li>
  <li><a href="#">3</a></li>
  <li><a href="#">4</a></li>
  <li><a href="#">Next <span class="fa fa-chevron-
    right"></span></a></li>
</ul>
```

> 对于 Next 和 Prev，原来的标记中已经应用了类为 a-chevron-left 和 a-chevron-right 的 span 标签，以使用 Font Awesome 图标。

结果就是得到如下所示的屏幕截图：



(7) 下面让分页导航在网格下方居中。首先，把它包围在一个 div 标签中，给这个标签一个 row 类以保证它清除上方内容，然后添加一个恰如其分的自定义类 pagination-wrap：

```
<div class="row pagination-wrap">
  <ul class="pagination">
    <li> ...
  </ul>
</div>
```

(8) 现在就要自己写样式让这个组件在它的父元素内居中。在第 4 章中，我们使用了自定义的 LESS 文件_page-contents.less 可保存自定义样式。在此，我们创建一个更具体的文件来保存针对商品网格的样式。创建新文件_products-grid.less，将其与其他自定义 LESS 文件保存在一起，然后加入以下样式：

```
.pagination-wrap {
  text-align: center;
}
```

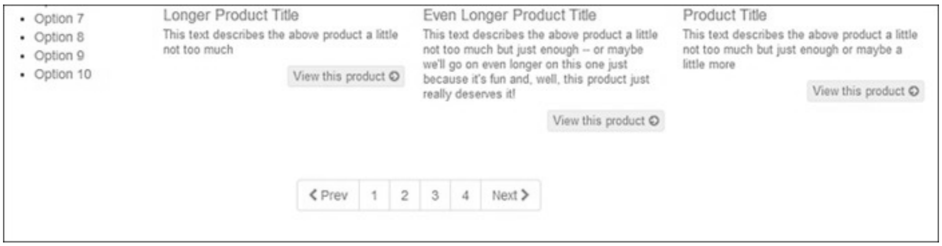
保存文件。

(9) 最后把新文件添加到 LESS 文件的导航列表。打开 less 文件夹中的_main.less，在注释 // Other custom files 下面添加一行导入语句：

```
@import "_products-grid.less"; // added
```

保存这个文件，然后编译为 CSS。

刷新浏览器，应该看到分页导航居中了。



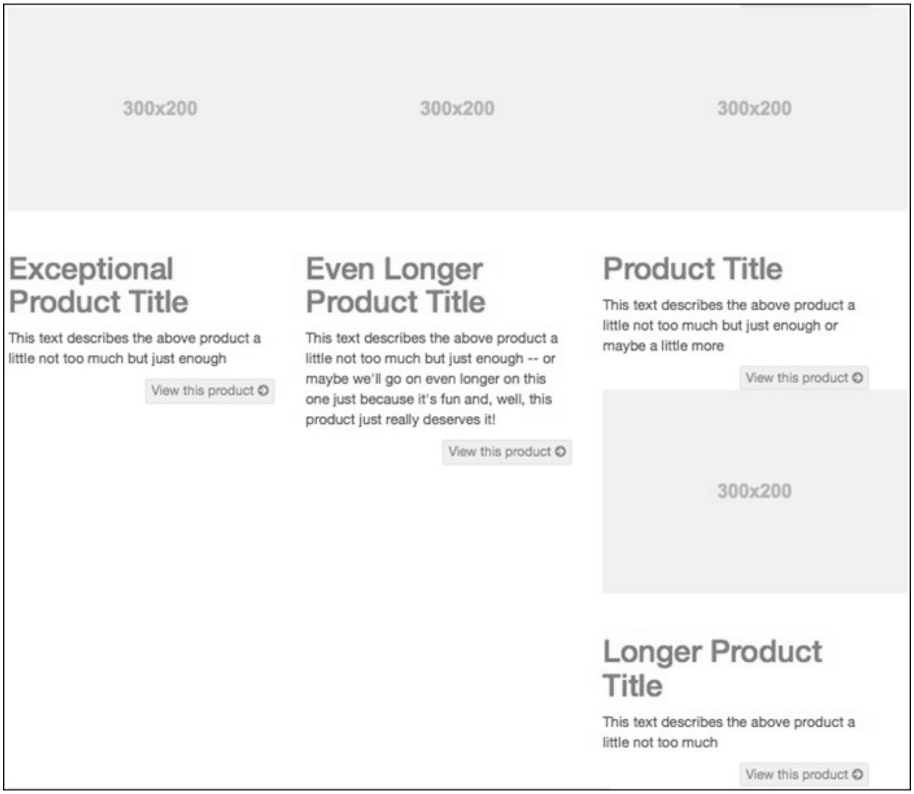
5.3　调整商品网格

这一节我们要把商品网格调整到位。仔细看一看每个商品的标记，你会发现它们都有一个类：col-sm-4：

```
<div class="product-item col-sm-4">
```

这个类虽然起到了约束每个商品宽度的作用，但整个网格看起来仍然尽不如人意。

主要问题是每个商品的高度不确定。因此，Bootstrap 在向左浮动所有商品时，后面的商品就有可能插入到前面的商品中。结果就是整个网格显得很乱，如下图所示。



目前，在中大型视口中，第 4~7 个商品由于高度不等，浮动后没有对齐。

我们的任务就是要调整网格系统，让所有网格的视觉效果得到增强。调整之后，马上就解决布局的问题。

(1) 因为要写自定义的样式，所以还要用编辑器打开_products-grid.less。

(2) 下面写一些样式，调整图片宽度、字号、内边距和外边距，代码如下：

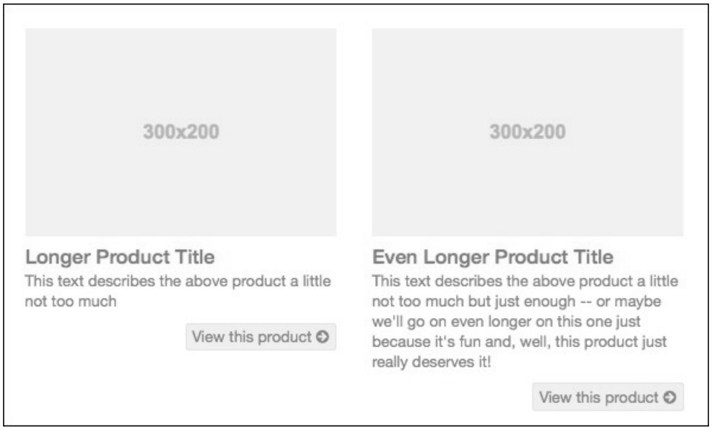
```
.product-item {
  padding-bottom: 32px;
  img {
    width: 100%;
  }
  h2 {
    font-size: @font-size-large;
    line-height: 1.2;
```

```
      padding: 0 !important;
      margin-top: 6px;
      margin-bottom: 2px;
      }
    p {
      font-size: @font-size-small;
      line-height: 1.3;
      color: @gray;
    }
  }
```

(3) 以上样式的作用如下：

- 给每个商品底部添加一些内边距；
- 把每个商品小图的宽度限制在商品区域内；
- 把 h2 的字号减小到@font-size-large；
- 把 p 的字号减小到@font-size-small；
- 把 h2 的内边距减小为 0，使用!important 保证覆盖我们在标准页面中应用的冲突规则；
- 把 p 的字体颜色设置为@gray。

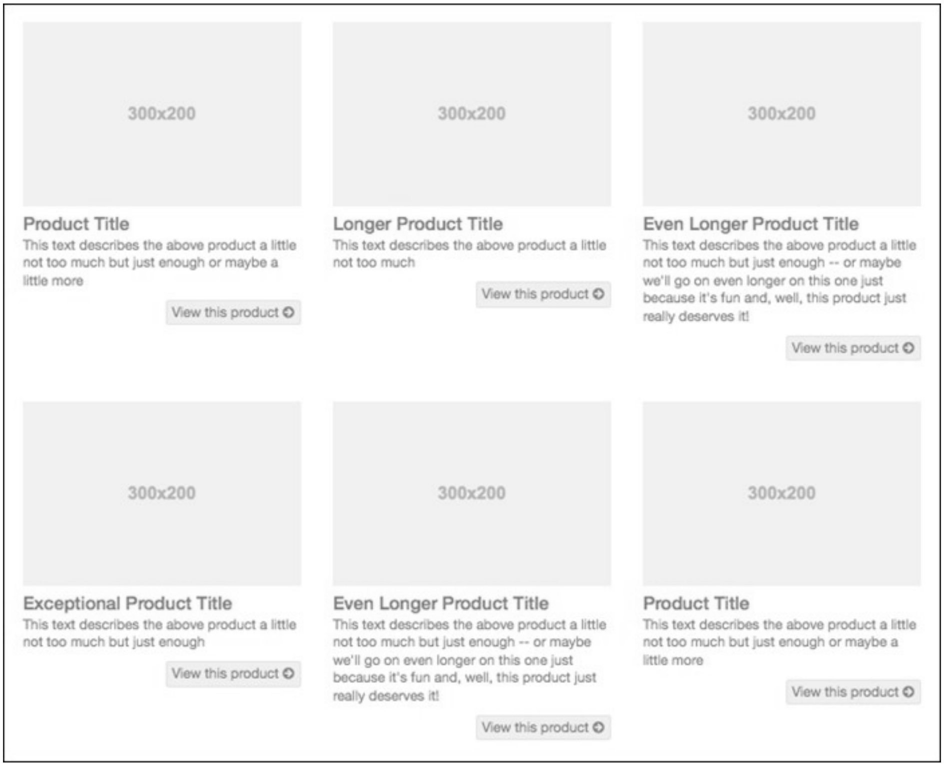
保存新样式，编译为 CSS，再刷新浏览器。尽管此时的布局还不整齐，但整体效果已经大为改观，如下图所示：



(4) 现在该来解决布局问题了。解决问题的关键是找到最高的商品。假设我们有一个指南，其中对每个商品使用什么图片和文字介绍都有规划。所有商品的小图都是标准大小，文字说明也不会比当前我们示例页面中的多。这样的话，我们就可以给所有商品都设置一个固定的高度，或者使用 em 或 ex 等更灵活的单位。在我们这个练习中，我们就使用 360px 的固定值，并把超出的部分隐藏起来。因为这些规则针对的是页面布局，所以我们要单独写一条规则，文件当然还是_products-grid.less:

```
.product-item {
  height: 360px;
  overflow: hidden;
}
```

保存这个文件，编译为 CSS，然后刷新浏览器。你会发现布局问题瞬间得到了解决！结果如下图所示：



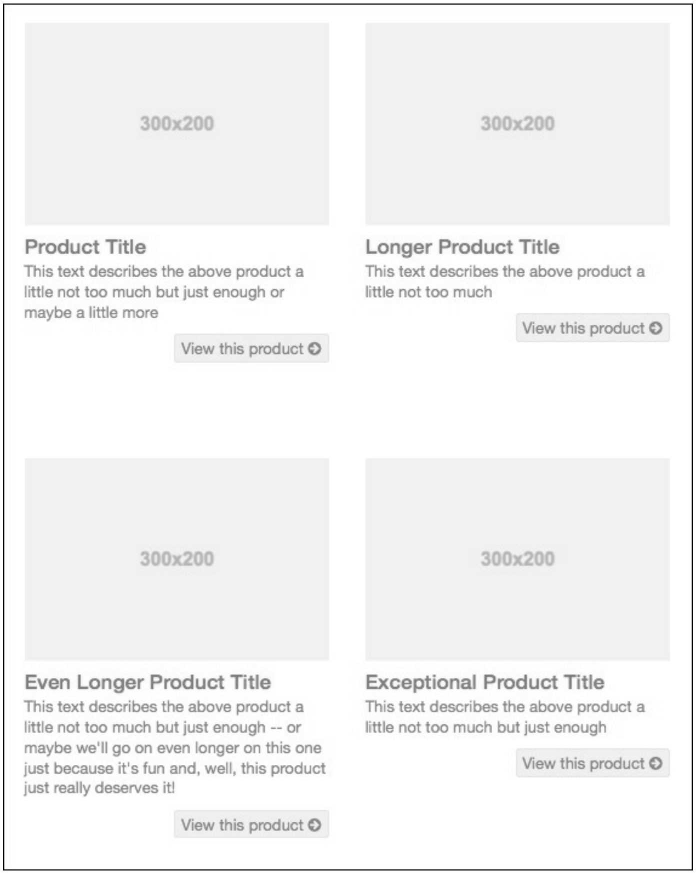
(5) 在此之后，我们就可以放心地使用 Bootstrap 的响应式分栏类，去调整不同视口的布局效果了。具体来说，我们希望当视口小和超小时，每行只显示两个商品；而当视口中等或较大时，每行显示三个商品。为实现这个效果，我们要找到并替换每个商品中的类，结果要变成如下所示：

```
<div class="product-item col-xs-6 col-md-4">
```

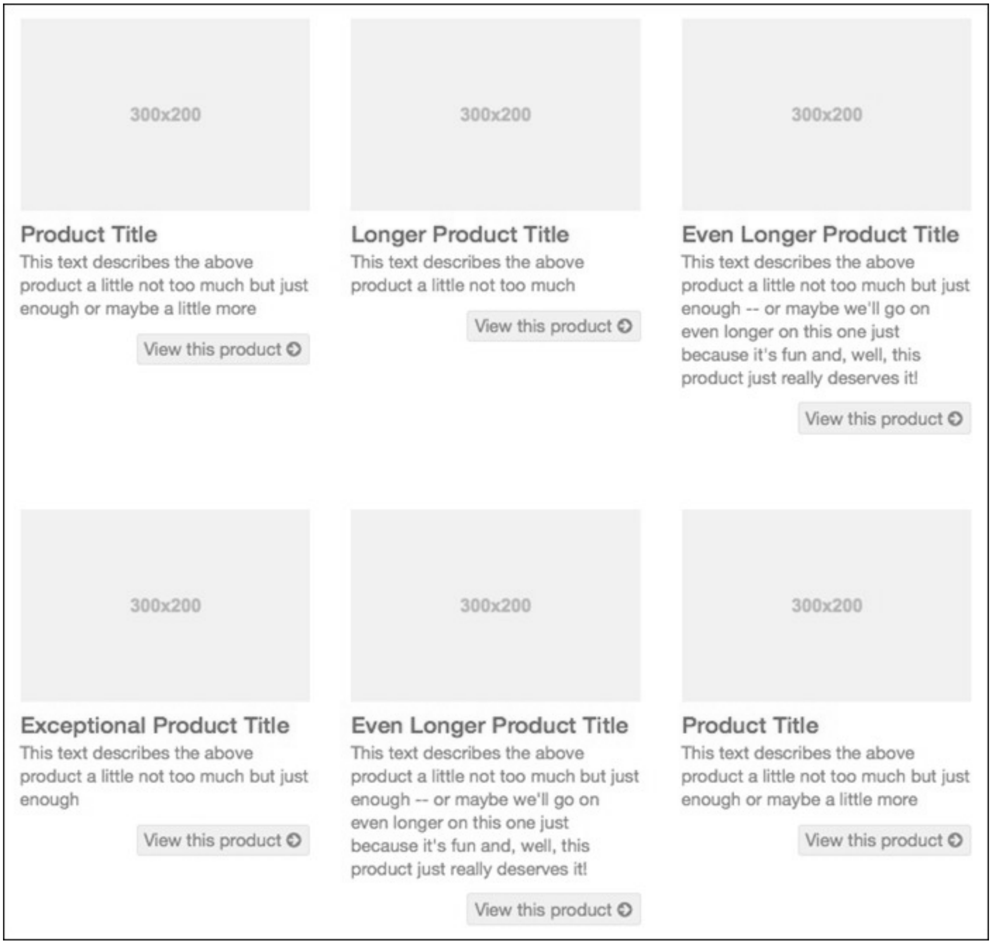
替换成这两个类之后，每个商品在超小和小视口中将会是屏幕宽度的一半，而在中大视口中将切换成屏幕宽度的三分之一。

保存，编辑并刷新浏览器。拖放浏览器窗口，变小再变大，看看商品动态变化的情况。

以下是小和超小视口下的情景：



然后，在中大视口中，商品会变成三栏：

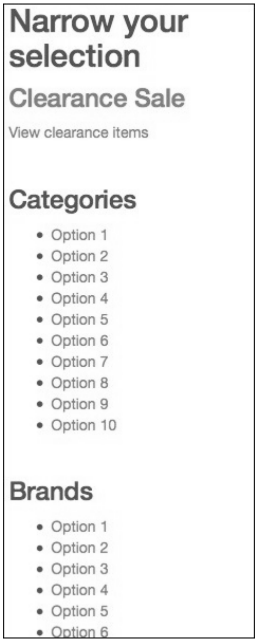


看着真是一种享受啊。

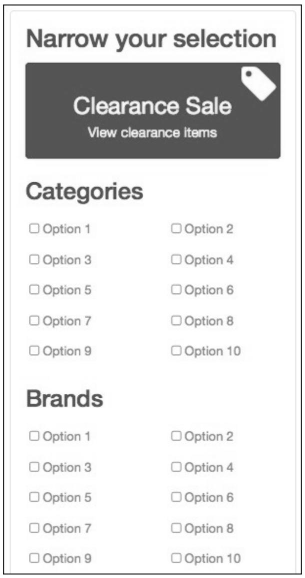
接下来我们来设计筛选选项及侧边栏。

5.4 侧边栏和筛选选项

侧边栏和筛选选项就在商品标记的前面。在小、中、大视口中，侧边栏目前都位于左侧。目前侧边栏的样子如下所示：



而在完成设计工作后，我们希望把 Clearance Sale 做成一个超大按钮，把筛选选项排成两栏，而且每个选项前是复选框而非项目符号，如下图所示：



下面先从基本的样式开始，把布局弄好。

5

5.4.1　基本布局

我们先来调整字体、颜色、外边距和内边距。

在_prod ucts-grid.less 中添加如下规则：

```
.grid-options {
  .panel;
  .panel-default;
  padding-top: 12px;
  padding-bottom: 24px;
  > h2 {
    margin-top: 0;
    font-size: 1.5 * (@font-size-large);
    line-height: 1.2;
    color: @gray-dark;
  }
}
```

上面的代码用途如下：

- 给侧边栏应用 Bootstrap 默认的 `panel` 样式（参见：http://getbootstrap.com/components/#panels）；
- 给侧边栏添加上、下内边距，让新背景贯穿侧边栏内容区；
- 调整 `h2` 标题的字号、行高和颜色。

下一步来做 Clearance Sale 按钮。

5.4.2　Clearance Sale按钮

我们要把 Clearance Sale 链接变成一个超大的吸引人的按钮。

按照下面的说明调整标记：

- 把链接的标题和段落都转换成按钮；
- 添加自定义的按钮类 `btn-feature`，这是我们在第 4 章创建的，具有特殊的颜色——红色；
- 给整个标签添加 Font Awesome 图标，通过使用 Font Awesome 内置的 `icon-3x` 类，将图标放大三倍。

> 要了解 Font Awesome 特殊尺寸类的更多信息，请参考相关文档：http://fontawesome.io/examples/#larger。

调整后的标记如下所示：

```
<a class="btn btn-feature choose-clearance" href="#">
  <span class="icon fa fa-tag fa-3x"></span>
```

```
  <h3>Clearance Sale</h3>
  <p>View clearance items</p>
</a>
```

效果立竿见影，我们向目标迈进了一大步：



下面再细化，完成下列目标。

(1) 将 Clearance Sale 显示为块级元素，使用`.center-block()`这个 Bootstrap 的混入将其居中。
(2) 强制其宽度为包含栏的 92.5%。
(3) 添加上、下内边距。
(4) 覆盖 Bootstrap 按钮的 `white-space: nowrap` 规则，让文本可以折行（Bootstrap 的 `white-space` 规则是在 less/bootstrap/buttons.less 中定义的，关于这个属性的更多信息，大家可以参考 http://css-tricks.com/almanac/properties/w/whitespace/ ）。
(5) 将按钮设置为相对定位，以便对标签应用绝对定位。
(6) 调整标题和段落的字体、颜色和外边距。
(7) 把标签图标定位到右上角。

以上目标通过下列规则就可以实现：

```
.choose-clearance {
  .center-block();
   width: 92.5%;
  padding-top: 20px;
  padding-bottom: 12px;
  white-space: normal;
  position: relative;
  h3 {
    font-weight: normal;
    color: #fff;
    padding-top: 4px;
    margin: 6px;
  }
  p {
    margin: 6px 20px;
    line-height: 1.2;
  }
  .icon {
```

```
    position: absolute;
    top: 0;
    right: 2px;
  }
}
```

结果就像下面屏幕截图一样：



不止如此，这些样式在不同视口下的表现也很出色。不信就花点时间多试试。当然，有什么不满意的地方，请大家以此为起点自行改进。

接下来该轮到筛选商品的选项了。

5.4.3　选项列表

本节，我们要把几个列表转换成筛选选项。

如果花点时间分析一下在线商店 Amazon（http://www.amazon.com）或 Zappos（http://www.zappos.com）的商品筛选选项，会发现这些选项其实是链接列表，而且每个选项都被搞成了复选框的样子。我们也要把链接做成复选框的样式，用户只要选择就会勾选，另外我们还要调整它们以适应多样化的设备，包括平板电脑和智能手机。

> 在 Amazon 和 Zappos 等电子商务网站上，筛选项与内容管理系统是关联的，网格中的商品会随着用户选择筛选项而动态变化。Bootstrap 是一个前端设计框架，不是内容管理系统。因此，我们这个练习做不到动态筛选商品。但我们这个页面完成后，是完全可以在内容管理系统中使用的。

先从每个列表的 h3 元素开始，我们调整它们的大小、行高、外边距和颜色：

```
.grid-options {
  > h3 {
    font-size: @font-size-large;
    line-height: 1.2;
    margin-top: 12px;
    color: @gray-dark;
  }
}
```

> 这里要使用>h3 子选择符，因为我们不希望它应用到 h3 标签，特别是不能应用到 Clearance Sale 按钮中的 h3 标签。

好，再把注意力集中到无序列表上。它们都有一个特殊的类，叫 options-list，我们就用它作为选择符，确保只针对这些特殊的列表。

首先去掉项目符号和内边距：

```
.grid-options {
  ..
  .options-list {
    list-style-type: none;
    padding-left: 0;
  }
```

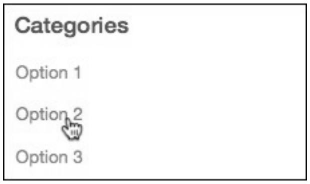
接下来是链接样式。稍后我们还要给列表项添加样式，因此我们把这些样式包含在了嵌套的选择符中。

```
...
  li {
    a {
      .btn;
      .btn-sm;
      padding-left: 0;
      padding-right: 0;
      color: @gray;
      &:hover,
      &:focus,
      &:active,
      .active & {
        color: @link-color;
    }
  }
}
```

以上规则的作用如下。

- 我们利用 LESS 通过.btn 类加入了基本的按钮样式，包括显示 inline-block 链接和额外的内边距：
  - 因为没有添加其他按钮类，所以也没有出现背景颜色；
  - 通过添加基本的按钮样式，可以让用户更方便点击，使用手指或鼠标皆宜。
- 我们再通过.btn-sm 类引入相关样式，以减少内边距，并让字号比标准按钮再小一些（要了解 Bootstrap 的按钮类，请参见：http://getbootstrap.com/css/#buttons）。
- 接着删除无序列表的左和右内边距。
- 再把链接文本的颜色改为@gray。
- 最后，设置悬停、焦点和活动链接的颜色为@link-color。

现在应该保存、编译并刷新浏览器，看看结果。结果应该如下图所示：



每个选项链接都有了内边距，字号和颜色也都合适了。

> 有读者可能奇怪为什么我要在这里借用按钮的`.btn`和`.btn-sm`类，而不是直接把这两个类写进标记。当然也可以那么做，但考虑到链接的数量那么多，我想还是通过 CSS 来应用样式更便捷。本章后面几节，我们将继续沿用这种思路，对 Font Awesome 图标样式也采用 LESS 而非直接在标记中添加类来应用。

好了，下一步是给选项链接添加复选框。

5.4.4　为选项链接添加Font Awesome图标复选框

本节，我们将使用 Font Awesome 图标在选项链接左侧添加复选框。但我们不在标记中添加，而是在 LESS 中添加，因为更快捷。然后我们更进一步，加入另一个 Font Awesome 图标，以表示悬停、聚焦和活动状态下勾选的复选框。

通过 LESS 添加图标需要从三个文件中取得 Font Awesome 样式。首先，从 font-awesome 文件夹的 core.less 中获得基本的样式。在这个文件中，可以看到以下重要的样式：

```
.@{fa-css-prefix} {
  display: inline-block;
  font-family: FontAwesome;
  font-style: normal;
  font-weight: normal;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
```

以上样式是所有 Font Awesome 图标样式的基础，包括作为字体的 Font Awesome 图标，以此为基础可以进一步加强相应的样式。

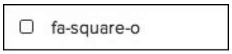
对现在的需求来说，我们不需要选择符也不需要花括号，只需要其中的规则。我们要把这些样式应用给选项链接。最重要的，我们要使用`:before`伪元素，因为可以确保结果最佳。

要了解更多 CSS2.1 :before 伪元素的信息，请参考这篇文章：http://coding.smashingmagazine.com/2011/07/13/learning-to-use-the-before-and-after-pseudo-elements-in-css/。

从 core.less 中复制上面的规则（不包括选择符），粘贴到_products-grid.less 文件中，嵌套如下：

```
.grid-options {
   ...
   li {
      ...
      a {
         ...
         &:before {
           // from font-awesome/core.less
           display: inline-block;
           font-family: FontAwesome;
           font-style: normal;
           font-weight: normal;
           line-height: 1;
           -webkit-font-smoothing: antialiased;
           -moz-osx-font-smoothing: grayscale;
```

这些规则为我们下一步打下了基础。下一步就可以指定使用哪个 Font Awesome 图标了。浏览这个页面：http://fontawesome.io/icons/，会发现空复选框的图标和类名：



这个图标的 LESS 规则可以在 font-awesome 文件夹的 icons.less 文件里找到。打开该文件，搜索字符串}-square-o（包括右花括号可以减少匹配项，从而缩小范围），可以看到下面这一行：

```
.@{fa-css-prefix}-square-o:before { content: @fa-var-square-o; }
```

对于前面这一行，我们只需要 content: @fa-var-square-o。把它复制粘贴到之前&:before 选择符中规则的后面：

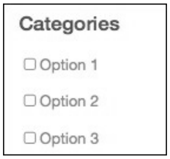

```
li {
  ...
  a {
    ...
    &:before {
      ...
       content: @fa-var-square-o;
```

最后，我们想取得另一些 Font Awesome 样式，为图标设置固定的宽度，避免图标在切换时出现位移。这些样式可以在 font-awesome 文件夹的 fixed-width.less 文件中找到。复制

下面这两行，同样粘贴到&:before 选择符中：

```
width: (18em / 14);
text-align: center;
```

添加上面的样式后，编译它们为 CSS 并刷新浏览器。应该看到下图所示的复选框：



接下来，我们以同样的方式添加选择符和规则，把 Font Awesome 复选框图标的勾选版应用给链接的悬停、焦点和活动状态：

```
li {
  ...
  a {
    &:before {
     ...
    content: @fa-var-square-o;
    }
    &:hover:before,
    &:focus:before,
    &:active:before,
    .active &:before {
      content: @fa-var-check-square-o;
    }
   }
```

保存文件，编译为 CSS，然后刷新浏览器。当鼠标悬停在某个链接上时，就会看到被勾选的复选框，如下图所示：



> 请大家注意，目前我们还没有办法强制某个链接停留在活动状态，因为我们没有内容管理系统支撑。但我们的样式已经齐备，如果有了内容管理系统，就可以直接用了。

就这样了！我们成功地给链接添加了复选框，能对用户的操作给出反馈了。

接下来，考虑一下充分利用侧边栏的空间，让选项浮动起来。

5.4.5 使用LESS混入在栏中对齐选项

上一节，我们使用 LESS 实现了以往需要通过添加标记实现的功能。考虑到筛选项的数量很多，这样做效率最高。同样的思路也适用于我们对齐侧边栏中的选项。

当然，如果使用 Bootstrap 的 `row` 和栏类，通过调整标记也是可以的：

```
<ul class="options-list options-categories row">
     <li class="col-xs-6"><a href="#">Option 1</a></li>
     <li class="col-xs-6"><a href="#">Option 2</a></li>
     ...
```

但有了 Bootstrap 的混入，我们用几行 LESS 代码可以就实现同样的效果。

(1) 首先给`.options-list` 选择符应用`.make-row()`混入：

```
.options-list {
   .make-row();
    ...
```

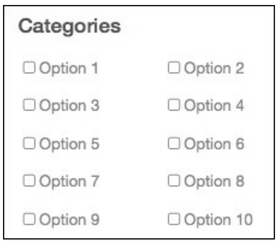
这个混入加入的样式与我们在标记中添加 `row` 类加入的样式一样。但这里只需要一行代码。

(2) 然后使用`.make-xs-col()`混入给列表项应用分栏规则：

```
li {
  .make-xs-column(6);
```

这样就跟我们给相关的 `li` 标签添加 `col-xs-6` 类的效果一样了。

(3) 添加前面两行之后，保存文件，编译为 CSS，再刷新浏览器。应该可以看到选项链接分成两栏了：



不错吧！

接下来针对小视口进行一番调整。

5.4.6 针对平板和手机调整选项列表布局

我们要限制选项面板的宽度，让它在平板电脑中不至于太宽。

目前来看，Clearance Sale 按钮有点太宽了。在 480~768px 下，选项列表相隔也太远了。相应的截图如下所示：



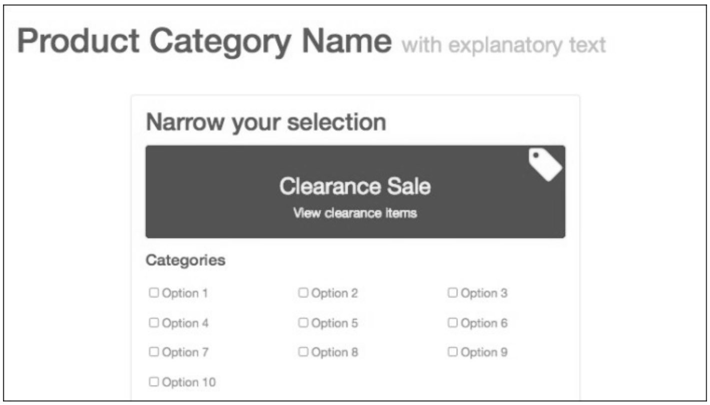
其实只要给选项面板设置一个最大宽度就行了：

```
.grid-options {
   ...
  max-width: 480px;
```

下面我们再调整选项列表，让它们在小视口中显示为三栏。使用 LESS，可以在适当的选择符中嵌套一个媒体查询，然后在其中添加一个用于调整的`.make-xs-column(4)`混入：

```
li {
   .make-xs-column(6);
   @media screen and (max-width: @screen-xs-max) {
     .make-xs-column(4);
   }
```

这样调整之后，保存文件，编译为 CSS，然后在小屏幕（视口）中测试一下。应该看类似下图所示的结果：



现在再我们解决下一个问题：在单栏布局中隐藏筛选选项，只在需要时显示。

5.4.7　在手机上折叠选项面板

现在，筛选项占据了相当多的垂直空间。这在小屏幕上是个问题，会把商品网格推到页面下方很远的地方。



原因就是筛选项不必要地占据了大量空间。商品本身才是最应该首先显示的。我们既要让用户迅速看到商品，也可以在需要时打开筛选项。

为此，我们使用 Bootstrap 的折叠插件。下面几步讲解如何对选项面板使用折叠插件，同时添加一个扩展面板的按钮，并把折叠行为限定在窄视口中。

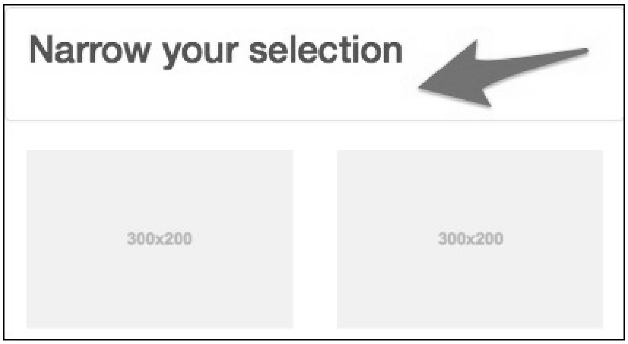
(1) 在编辑器中打开 products.html。

(2) 添加一个新的 `div` 标签，包装 Clearance Sale 按钮和三个选项列表。给这个 `div` 添加一个特殊的类 `collapse`，以及一个唯一的 ID，以便 JavaScript 插件找到它，同时也添加一个同名的类：

```
<div id="options-panel" class="options-panel collapse">
    <a class="btn btn-feature choose-clearance" href="#">
    ...
    <h3>Categories</h3>
    <ul class="options-list options-categories">
        <li><a href="#">Option 10</a></li>
        ...
    </ul>
</div><!-- /#options-panel.collapse -->
```

> Bootstrap 的折叠 JavaScript 插件也是我们在响应式折叠导航条中使用的。这个插件也可以用于其他用途，具体可参考 Bootstrap 的文档：http://getbootstrap.com/javascript/#collapse。

(3) 保存文件，刷新浏览器。你会发现 Clearance Sale 按钮和选项列表顿时从眼前消失了。只剩下选项面板上方的 `h2` 标题“Narrow your selection”了，如下图所示：



现在需要一个切换按钮，在被点击时显示筛选项。

(4) 在这个还能看见的内容为“Narrow your selection”的 `h2` 中，添加一个 `button` 元素，还有相应的属性：

```
<h2 class="clearfix">Narrow your selection
  <button type="button"
  class="options-panel-toggle btn btn-primary pull-right"
    data-toggle="collapse" data-target="#options-panel">
```

```
  <span class="icon fa fa-cog fa-2x"></span>
  </button>
</h2>
```

下面来解释一下上面的标记：

- 给 h2 添加的 clearfix 类可以确保它包含切换按钮，因为切换按钮是浮动到右边的（可以在 bootstrap 文件夹的 utilities.less 文件中找到 clearfix 类，在 mixins.less 文件中找到生成它的混入）；
- 类 btn 和 btn-primary 会给新的按钮添加 Bootstrap 的基本按钮样式，背景颜色为@brand-primary；
- 类 pull-right 会把按钮浮动到右侧；
- 在 button 元素中，我们放了一个 Font Awesome 齿轮图标，使用 fa-2x 类放大到两倍。

保存并刷新浏览器，可以看到下面的结果：



(5) 下面要写一些规则，在中大屏幕中隐藏切换按钮并展开选项面板。为此，可以在 _products-grid.less 中添加以下规则：

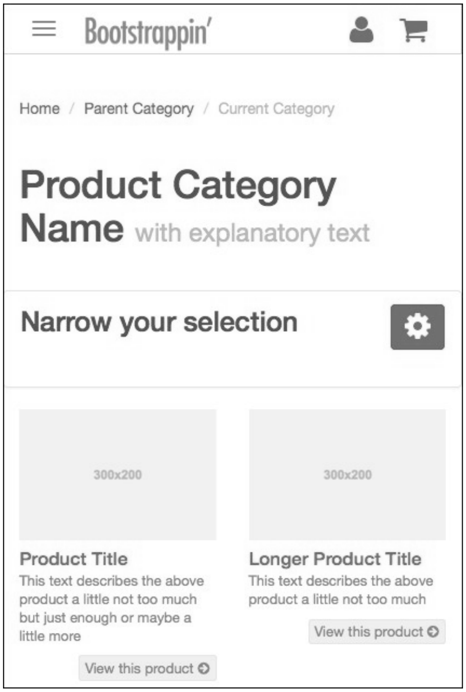
```
// Responsive adjustments
@media (min-width: @screen-sm-min) {
    .options-panel {
        display: block;
    }
     .options-panel-toggle {
     display: none;
    }
}
```

(6) 这些规则的作用如下：

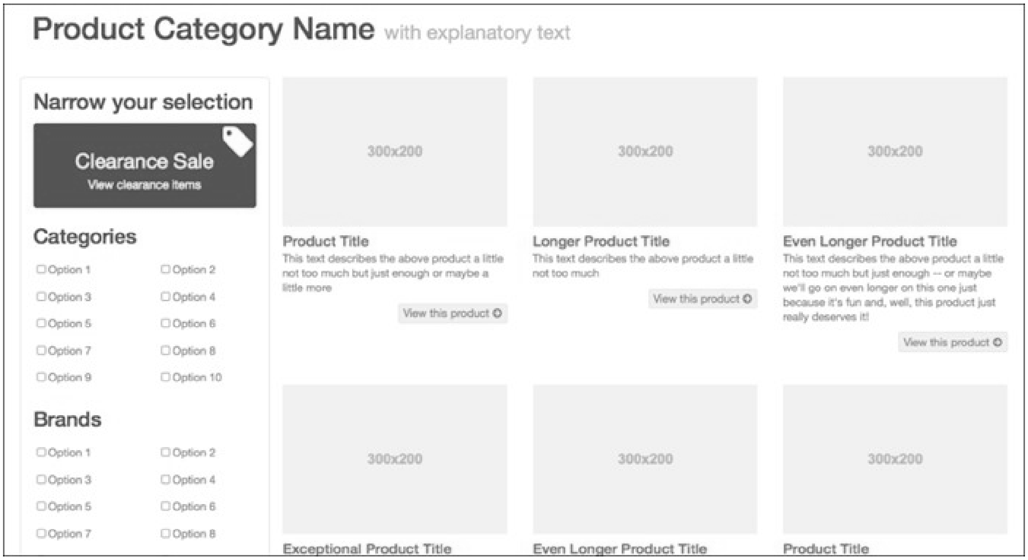
- 媒体查询保证只把规则应用到小中大视口；
- 第一条规则抵消 collapse 类的作用，它默认是隐藏元素的；
- 第二条规则隐藏切换按钮。

保存并刷新，应该就能看到想要的结果了。

在超小视口中，选项列表会折叠起来，但切换按钮可见：



在小、中、大视口中，切换按钮隐藏，选项列表可见：



祝贺你，终于完工啦！

5.5 小结

本章，我们做了以下事情。

- 利用 Bootstrap 的样式快速实现了面包屑、页面标题和分页导航，并根据需要进行了定制；
- 调整了 Bootstrap 的网格样式，为商品创建了整齐的布局，关键是所有商品的高度要一致；
- 为复杂的 Clearance Sale 按钮应用了样式，用到了`@brand-feature`这个红色背景；
- 利用`btn`类的样式让筛选项更容易点击或触摸，通过自定义满足了我们的特殊需求；
- 使用 Bootstrap 的分栏类，加上响应式调整，对齐了筛选项列表，而且适合多视口；
- 在自定义样式表中借用 Font Awesome 样式在筛选项旁边添加复选框；
- 设置了选项面板在窄视口中折叠，在小中大视口中可见。

再次祝贺你！现在我们的企业网站集成了一个很像样的电子商务页面。

下一章，我们更上一层楼，创建一个单页营销网站。

6 单页营销网站

我们已经掌握了很多使用 Bootstrap 的重要技能。现在，是时候拿出更多的创意来帮助客户实现他们全方位在线营销的愿望了。本章将带领大家做一个漂亮的单页高端营销网站。

本章要完成以下任务。

- 一个大型介绍性传送带图片展示区，配有自定义的响应式欢迎信息；
- 一个客户留言区，显示为带标题的图片墙，就像砖垒的一样；
- 一个功能清单，使用大号 Font Awesome 图标；
- 一个带有自定义价目表的注册区；
- 一个带动态滚动的 ScrollSpy 导航条。

6.1 概况

有一位潜在客户联系我们，她深深地爱上一种漂亮的网站，就是那种可以垂直滚动，以强烈的视觉冲击力展示商品，最后还有一个突出的行动召唤按钮的单页网站。她想让你做一个。

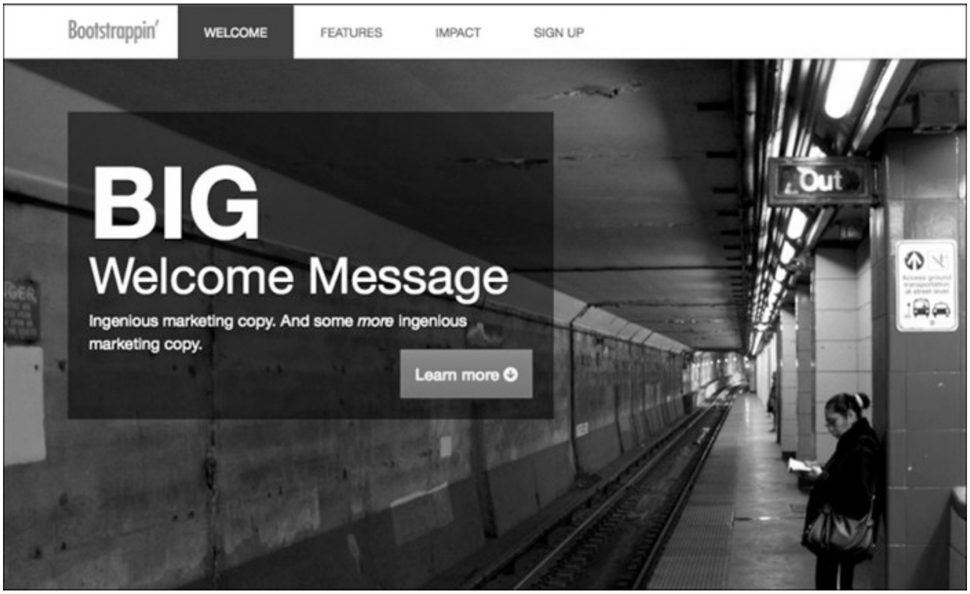
这位客户知识渊博、目光如炬。她经常光顾 http://onepagelove.com，并且收集了一堆最喜欢的功能，包括：

- 一个清新，具有现代美的网站；
- 一条介绍性的欢迎语，打在吸引人的背景图片之上；
- 一个高效的商品展示区，用醒目的图标来突出；
- 精致的客户留言板，深具视觉冲击力；
- 三个能让客户一目了然的价目表，方便选择，快捷注册；
- 不断沟通！一切都在吸引用户一步一步向下看，让人几乎无法拒绝点击最后的注册按钮。

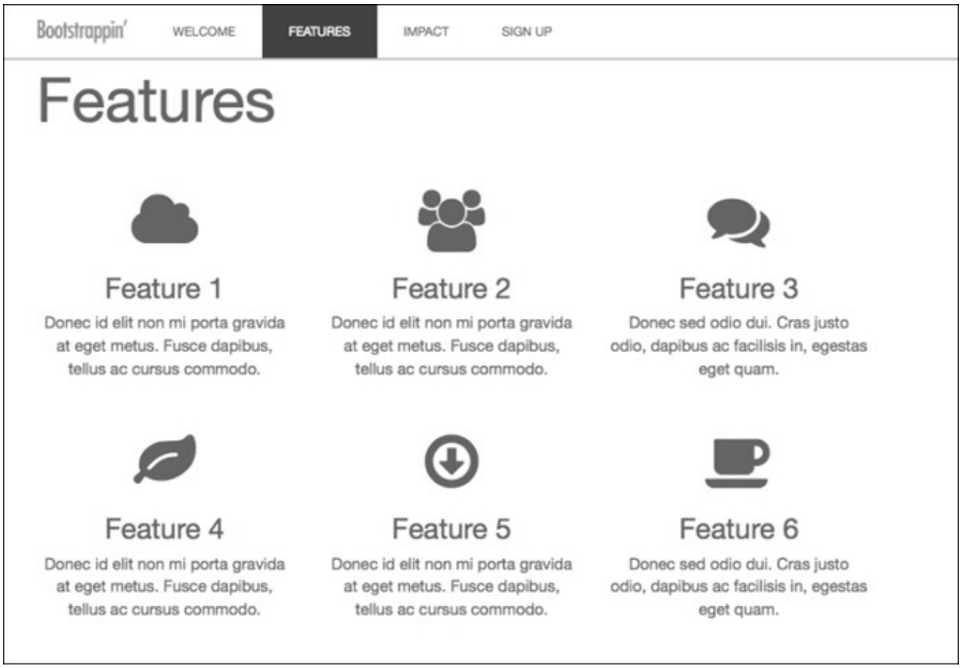
为了保持她未来商品的神秘感，我们的客户没有为我们提供实际的商品或服务图。她给了我们一个设计图，设计图中使用了占位图片。

第一部分将是一张横贯全屏的高清图片，上面有一条大大的欢迎语，以及一个邀请向下

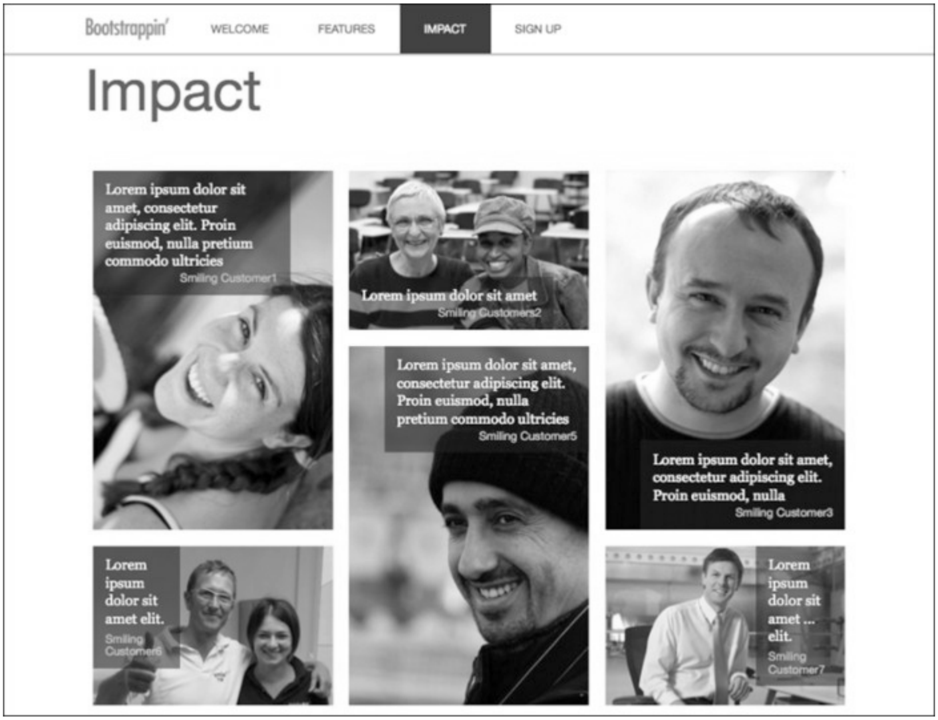
滚动阅读的按钮，如下图所示：



第二部分将列出商品的六个重要功能，分成三栏，并配备了相应的图标，如下图所示：



第三部分展示客户的赞誉，有图片，有文字，以图片墙形式呈现：



第四部分也是最后一部分，提供了三个可以选的方案，每个方案对应相应的报价，同时在视觉上突出中间的报价方案，如下图所示：



真是一位与时俱进的客户，所以她最后还要求我们的方案必须完美地适应平板电脑和智能手机。

目标很远大。当然没问题。动手吧。

6.2 初始文件

本章项目的起始文件位于 06_Code_Begin 文件夹中。与本书前几章的项目一样，文件的核心是 Bootstrap 3 LESS、JavaScript 和按照要求组织的标记，搭配了 HTML5 Boilerplate 和 Font Awesome 图标字体。

项目的文件夹和文件结构与前几章中的非常类似。下面我们就简单回顾一下 LESS 文件。

- 默认的 Bootstrap 文件位于/less/bootstrap/文件夹。
- Font Awesome 图标字体的 LESS 文件位于/less/font-awesome/。
- 我们自定义的 LESS 文件就在 less 文件夹中，以下划线开头，一眼就能看出来。自定义的 LESS 文件如下。
  - __main.less：这是导入所有其他文件的主文件，应该把它编译为 css/main.css；
  - _variables.less：这个文件基于 Bootstrap 定义的变量新增了一些变量；
  - _navbar.less：这个文件包含导航条的自定义样式；
  - _page-contents.less：这个文件包含页面内容区的自定义样式；
  - _footer.less：这个文件包含页脚的自定义样式。

在前几章的项目中，我们已经用到过这些文件了。

但对于本章的项目，这些文件有一些不同之处。

- 我针对本章的项目对以下 LESS 文件作了一些改动。
  - _variables.less：调整了一些变量，特别是针对导航条的变量，我特意给出了注释说明；
  - _navbar.less：这里的样式用于限制站点 Logo 图，将它放置到视线的开始位置，并与整体网格保持一致。
- index.html 文件中的大部分标记都已就绪。
- 图片保存在 img 文件夹中，其中的图片已针对 Web 进行了缩放、裁剪和优化，也已经插入到了标记中适当的位置。

开始之前，大家可以先打开浏览器看看当前页面的样子。

6.3 了解页面内容

在浏览器中打开 index.html，可以看到下面列出的组件。当然，目前这些组件使用的都是

Bootstrap 的默认样式，稍后我们会添加自定义样式。

- 固定在顶部的导航条；
- 带一句大号欢迎语的高清图；
- 功能介绍，包括图标、标题、文字，分三栏；
- Impact 部分是成功用户的照片，占位文本代表他们的赞誉；
- Sign up Now!部分是三张价目表， 包括 Basic Plan、Premium Plan 和 Pro Plan，每个下面都有一个 Sign up Now!按钮；
- 页脚的 Logo；
- 图片出处（按照许可给出每张图片的来源）。

要查看标记，请用编辑器打开 index.html。在后面的几步中，我们逐渐熟悉这些标记。

6.4　调整导航条

本章的项目包含一个固定在顶部的导航条，链接在悬停和激活状态会有显著的颜色变化。为此，我通过设置相应的变量应用了一些样式。下面我来一一指出来，然后我们再探讨怎么调整标记。

如前所述，less/_variables.less 文件是以 Bootstrap 的 variables.less 文件为基础的。在这个文件里，我像前几章一样，修改了灰色变量。可以在文件一开头看到这些变量。

接着我又调整了导航条的变量，涉及它的高度、外边距、颜色、悬停颜色：

```
// Basics of a navbar
@navbar-height:                    56px;
@navbar-margin-bottom:             0;
...
// Navbar links
@navbar-default-link-color:        @navbar-default-color;
@navbar-default-link-hover-color:  #fff;
@navbar-default-link-hover-bg:     @gray;
@navbar-default-link-active-color: #fff;
@navbar-default-link-active-bg:    @gray-dark;
```

此外，也调整了导航条切换按钮的变量：

```
// Navbar toggle
@navbar-default-toggle-hover-bg:      transparent;
@navbar-default-toggle-icon-bar-bg:   @gray-lighter;
@navbar-default-toggle-border-color:  transparent;
```

最后，去掉了导航条切换按钮及其他元素的圆角。只要把三个`@border-radius-`变量的值改为 0 即可：

```
@border-radius-base:         0; // was 4px
@border-radius-large:        0; // was 6px
```

```
@border-radius-small:             0; // was 3px
```

除了这些自定义变量，我还稍微修改了一下_navbar.less。修改的地方主要是`.navbar-brand`的内边距，以便与Logo图片保持一定间距：

> 原来的代码被注释掉了，新添加的代码后面又加了注释。

```
.navbar-brand {
  float: left;
  // padding: @navbar-padding-vertical @navbar-padding-horizontal;
  padding: 12px 30px 0 15px; // to allow for logo image
```

此外，还自定义了导航条展开时的列表项，添加了左、右内边距，并把文字转换成大写：

```
// Uncollapse the nav
  @media (min-width: @grid-float-breakpoint) {
 ...
    > li {
      float: left;
      > a {
        padding-top: ((@navbar-height - @line-height-computed) / 2);
        padding-bottom: ((@navbar-height - @line-height-computed) / 2);
        padding-left: 24px; // added
        padding-right: 24px; // added
        text-transform: uppercase; // added
      }
    }
```

以上调整组合到一起，就得到了如下结果：



下面我们就从高清图和大号欢迎语开始。

6.5 定制高清图

本节，我们要自定义高清图，显示客户的大号欢迎语，同时要对标记进行一番调整。包括添加大背景图，放大欢迎语，然后调整其在多视口中的外观。

在index.html中，找到如下标记：

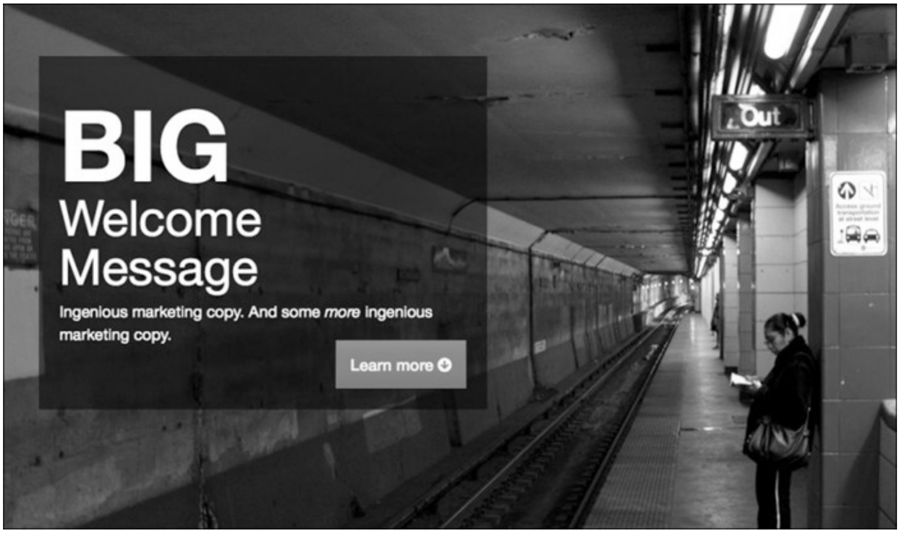
```
<!-- INTRO SECTION -->
<section id="welcome" class="jumbotron">
  <div class="container">
    <h1><strong>Big</strong> Welcome Message</h1>
    <p>Ingenious marketing copy. And some <em>more</em> ingenious
      marketing copy.<a href="#features" class="btn btn-lg btn-primary
```

```
    pull-right">Learn more <span class="icon fa fa-arrow-circle-down">
    </span></a></p>
  </div>
</section>
```

当前，在应用了 Bootstrap 默认样式后，结果如下图所示：



而在我们完成对高清图的调整之后，结果应该是这样的：



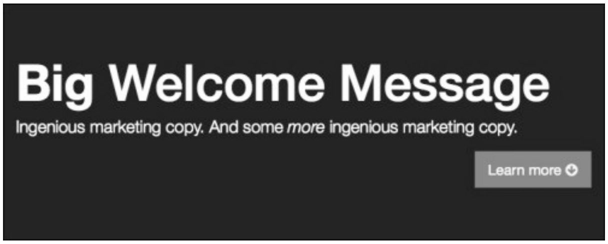
首先扩大显示区的高度，把高清图放进去。

(1) 在编辑器中打开自定义的 LESS 文件 less/_page-content.less。我们要用这个文件自定义很多页面的细节。

(2) 现在我们设置#welcome 部分的高度、背景颜色和字体颜色，同时也为按钮添加一些上外边距：

```
#welcome {
    height: 300px;
    background-color: #191919;
    color: #fff;
    .btn {
      margin-top: 16px;
    }
}
```

(3) 保存修改，编译为 CSS，刷新浏览器，应该可以看到如下结果：



接下来，我们使用媒体查询为中大屏幕添加背景图片（根据目前 Bootstrap 媒体查询默认的断点值，大屏幕指 991px 以上）。

> 如果愿意，也可以打开_variables.less，搜索并重温一下 Bootstrap 的媒体查询变量，比如`@screen-xs`、`@screen-sm`、`@screen-md` 和`@screen-lg`。

(1) 利用 LESS，可以在`#welcome` 的上下文中嵌套一个媒体查询。在这个媒体查询中，将 subway-906x600.jpg 指定为背景。对这里的断点来说，这张图片已经足够大了，但加载速度相对也很快：

```
#welcome {
  ...
  @media (max-width: @screen-sm-max) {
    background: #191919 url('../img/subway-906x600.jpg')
      center center no-repeat;
  }
}
```

(2) 保存文件，编译为 CSS，刷新浏览器。应该看到背景图片出现了，但只会在屏幕宽度为 911px 或更小的时候才会出现：



(3) 下面我们要扩展平板大小视口下高清图的高度。为此，要使用断点`@screen-sm-min`写一个媒体查询，把`#welcome`元素的高度变成 480px：

```
@media (min-width: @screen-sm-min) {
    height: 480px;
}
```

(4) 保存，编译，然后刷新浏览器。应该看到视口在 768~991px 之间时，高清图的高度会变成 480px，如下图所示：



(5) 接下来考虑中大（992px 以上）视口，此时把高清图变成 540px 高。在这个宽度下，就要使用更大的背景图片 subway-1600x1060.jpg,同时把`background-size`设置为`cover`：

```
@media (min-width: @screen-md-min) {
     height: 540px;
     background: #191919 url('../img/subway-1600x1060.jpg')
       center center no-repeat;
     -webkit-background-size: cover;
     -moz-background-size: cover;
     -o-background-size: cover;
     background-size: cover;
  }
```

(6) 有了这些样式，当视口变大时，就会显示 1600px 宽的背景图片了。而且在现代的浏览器，包括 Internet Explorer 9 中，背景图片还会拉伸以填满`#welcome`元素。

(7) 保存，编译，然后测试一下。没问题，所有断点基本都涵盖了。

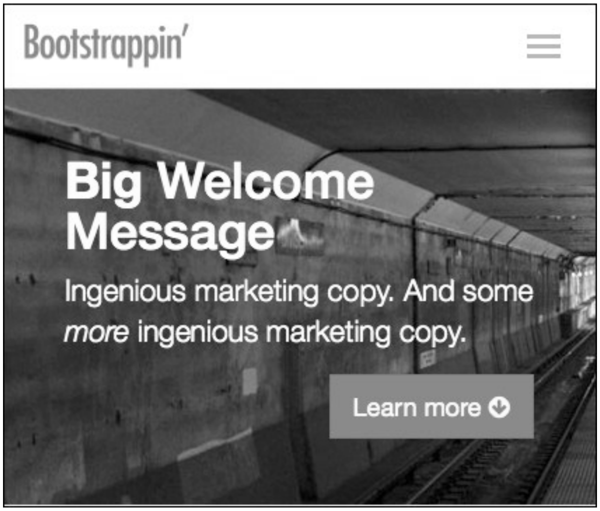
> 注意，在 Internet Explorer 8 中，当视口宽度超过 1600px 时，图片左右就会显示`#191919`的背景色。这种情况不会影响太多用户，但的确存在这个问题，只是问题不大而已。

接下来，我们为欢迎语添加样式，使其突出出来。

调整欢迎语

客户希望高清图上的欢迎语超级大。Bootstrap 的高清图样式把原字号增大了 1.5 倍，我们还要再增大一些。还要在宽屏幕中约束欢迎语的宽度，并在其下方衬托一个半透明的盒子。

目前的结果在超小屏幕中表现已经很好了：



不过，还是可以改进一些。那就是在文本底下衬托一个半透明的黑盒子。下面就来试试看。

(1) 在 index.html 中，高清图 `container` 类内部，添加一个新的类为 `welcome-message` 的 `div` 元素，包含 `h1` 标题和段落：

```
<section id="welcome" class="jumbotron">
  <div class="container">
    <div class="welcome-message">
      <h1><strong>Big</strong> Welcome Message</h1>
      <p>Ingenious marketing copy. And some <em>more</em> ingenious marketing copy.
        <a href="#features"
        class="btn btn-lg btn-primary pull-right">Learn more
        <span class="icon fa fa-arrow-circle-down"></span></a></p>
    </div><!-- /.welcome-message -->
  </div>
</section>
```

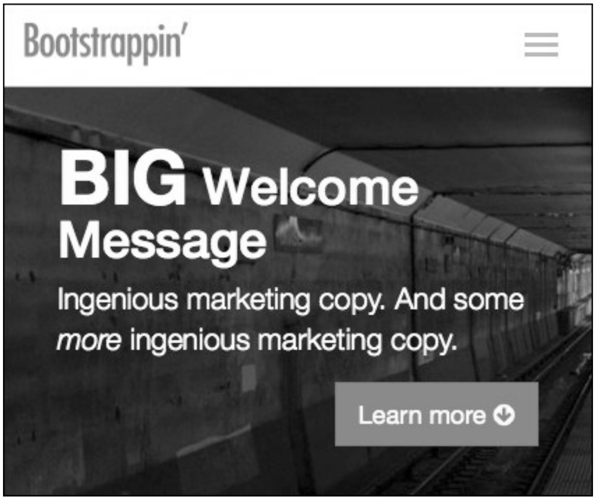
(2) 现在为这个 `div` 添加样式，分以下几步：

- 使用 HSLA 添加半透明黑色背景；
- 将其设为绝对定位，并通过将上、下、左、右设置为 0，将其拉伸至与高清图一样大小；
- 使用`#welcome` 将高清图设置为相对定位，以便确定欢迎语的位置；

- ❑ 给欢迎语添加内边距；
- ❑ 使用原有的 strong 标签把“Big”变成大写，同时增大字号。

```
#welcome {
    ...
     position: relative;
     .welcome-message {
       background-color: hsla(0,0,1%,0.4);
       position: absolute;
       top: 0;
       bottom: 0;
       left: 0;
       right: 0;
       padding: 30px 40px;
       strong {
         font-size: 1.5em;
         text-transform: uppercase
       }
    }
   ...
}
```

(3) 保存文件，编译 CSS，然后刷新浏览器。应该能看到背景变暗了，文本在这个深色背景上也更加引人注目，如下图所示：



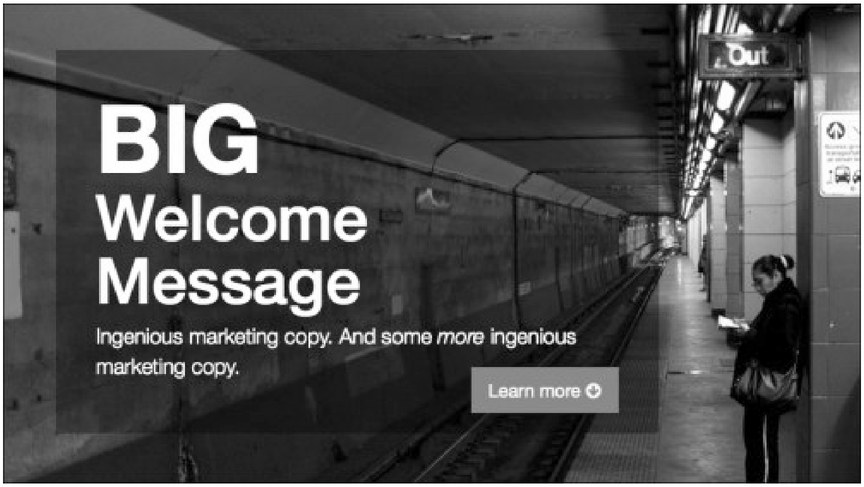
(4) 下一步，要考虑@screen-sm 断点。我们已经为这个断点写过媒体查询，当时是用来把高清图提高到 480px。就在这个断点内，我们要添加一些规则，完成以下任务：

- ❑ 把高清图的 container 设置为相对定位，使其成为新的定位参照点，以便我们从上方和左侧向内缩小欢迎语的盒子；
- ❑ 右侧向内缩小 20%；

- 将底边设置为 auto，以便盒子能收缩适应内容；
- 将“Big”设置为块级元素，单独显示在一行上。

```
@media (min-width: @screen-sm-min) {
    height: 480px;
    .container {
      position: relative;
    }
    .welcome-message {
      right: 20%;
      bottom: auto;
      strong {
        display: block;
      }
    }
}
```

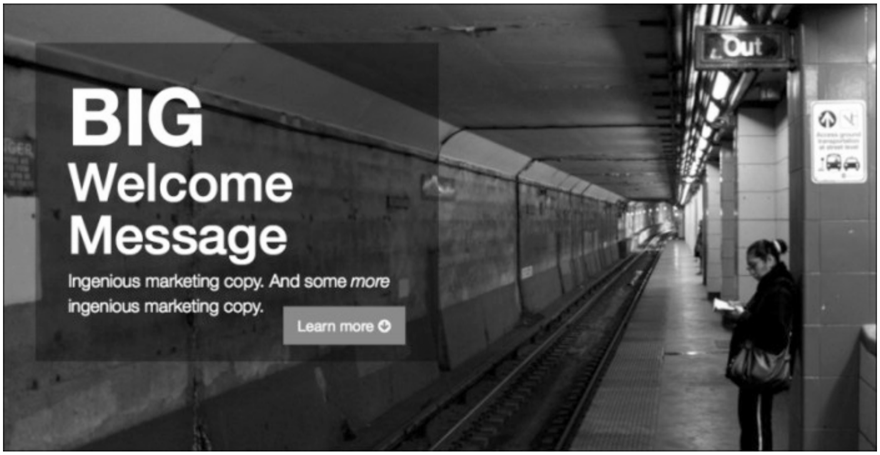
(5) 保存，编译并刷新浏览器。应该看到下图所示的结果：



(6) 最后，我们再针对中大视口作调整。在中大视口中，我们想限制一下欢迎语盒子的宽度。这次要用到之前针对断点@screen-md-min 创建的媒体查询：

```
@media (min-width: @screen-md-min) {
     ...
    .welcome-message {
       right: 50%;
     }
}
```

(7) 保存文件，编译到 CSS，然后刷新浏览器。在中大视口中应该看到下图所示的结果：



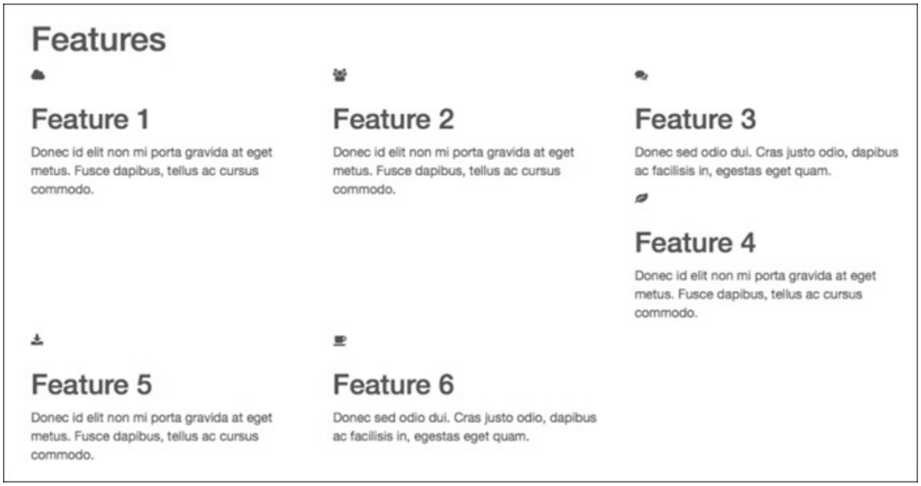
使命达成！

我们自定义的高清大图就此完成，满足了客户显示超大欢迎语的要求，同时还能适应平板、手机等设备的屏幕。关键是我们在此运用了移动设备优先的原则。

下面就是功能列表。

6.6　美化功能列表

包含图标、标题和简短文字描述的功能列表，目前看起来是这样的：



我们的目标是增大图标，居中对齐文本，然后平整网格而已。

看一下功能列表的标记结构：

```
<section id="features">
  <div class="container">
    <h1>Features</h1>
    <div class="row">
      <div class="features-item col-md-4">
        <span class="icon fa fa-cloud"></span>
        <h2>Feature 1</h2>
        <p>Donec id elit non mi porta gravida at eget metus. Fusce
          dapibus, tellus ac cursus commodo. </p>
      </div>
   ...
```

> 每个功能都有自己的图标、标题和段落，包含在自己的 div 标签中，这个标签有两个类：features-item 和 col-md-4。

知道了标记结构，接下来就可以写样式了。

(1) 在编辑器中打开的_page-contents.less 文件中，新开辟一块，并添加注释，表明是功能区的样式。

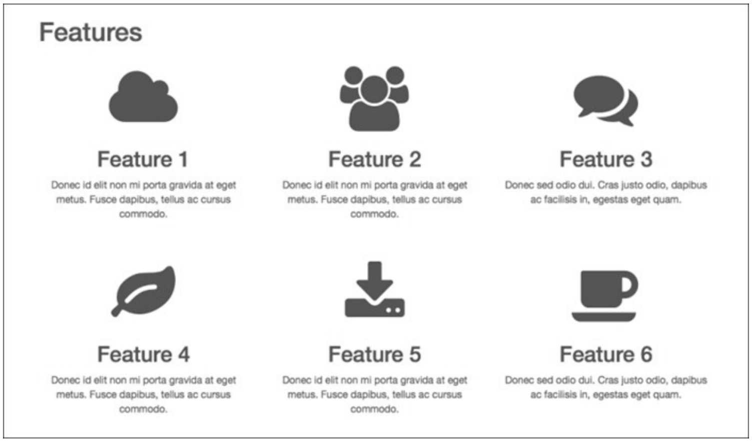
```
// Features Section
#features {

}
```

(2) 首先针对.features-item 部分，居中文本，添加内边距，并设定高度以避免浮动的功能项互相交错，同时将.icon 字体增大为 90px：

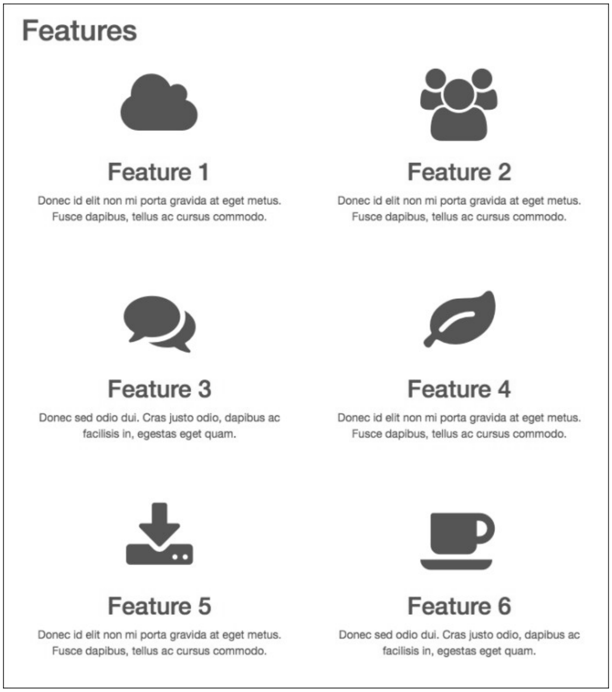
```
#features {
    .features-item {
    text-align: center;
    padding: 20px;
    height: 270px;
    .icon {
      font-size: 90px;
    }
  }
}
```

(3) 保存文件，编译为 CSS，然后刷新浏览器。在中等宽度视口中应该看到下图所示的效果：



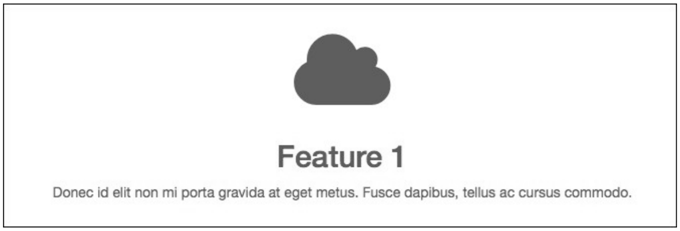
不错！

(4) 下面针对小视口调整功能列表。当前，每个`.features-item`都有类`col-md-4`，而我们希望在小屏幕中功能列表显示为下图所示的两栏，相应地要添加类`col-sm-6`：



(5) 再小一些，在超小视口中，功能项会自己变成一栏。

(6) 可是，在超小屏幕范围之上，即 500~767px 的时候，一栏的布局会导致文本描述太宽：



(7) 解决这个问题需要再添加一个媒体查询，为.features-item设置最大宽度，同时应用 Bootstrap 的.center-block()混入：

```
#features {
   .features-item {
      ...
      @media (max-width: @screen-xs-max) {
          max-width: 320px;
          .center-block();
       }
   }
}
```



在 bootstrap 文件夹中的 mixins.less 中，可以找到.center-block()。这个混入会对元素应用自动的左右外边距。

(8) 有了以上限制，.features-item元素在任何视口中都会保持理想的宽度了！



此时，我们又满足了客户对其网站这一部分的要求。下一步可以考虑用户评论区了。

6.7　装饰用户评论区

接下来的一部分叫“Impact”，用来展示成功用户的评论。在这一部分，我们看到的是成功用户的笑脸，还有他们对我们客户商品的赞美之词。这一部分开始的标记结构如下：

```
<!-- IMPACT SECTION -->
<section id="impact">
  <div class="container">
  <h1>Impact</h1>
     <div class="reviews">
```

每一条评论都像下面这样使用 `hreview` 微格式标记：

```
<div class="hreview review-item-1 thumbnail">
  <img src="img/smiling1-by-RomainGuy-600x900.jpg" alt="Customer Photo1">
  <div class="caption">
    <blockquote class="description"><p>Lorem ipsum dolor sit amet,
      consectetur adipiscing elit. Proin euismod, nulla pretium commodo
      ultricies</p></blockquote>
    <p class="reviewer">Smiling Customer1</p>
  </div><!-- /.caption -->
</div><!-- /.hreview -->
```

> 要了解 `hreview` 微格式的信息，请访问 http://microformats.org/wiki/hreview-examples。

为了方便布局和添加样式，我们使用了 Bootstrap 的 `thumbnail` 类结构，这个结构有以下好处：

- 在每条评论的父元素中，我们都会在 `hreview` 类旁边再添加一个 `thumbnail` 类；
- 评论内容，包括引用的话和评论者的名字，都包含在 `div class="caption"` 中。

这种缩略图（thumbnail）和说明（caption）结构对每条评论给出了整体封装。Bootstrap 的缩略图样式就是用来在我们期望的布局中限制图片和说明比例的。

当前这样的结构，无论从语义角度，还是从表达角度，都为我们提供了很好的基础。

我们知道，用户评论区最终要做成一面图片墙的样子，图片有横也有竖。为了让照片中的脸部都露出来，同时有地方叠加评论文字，我们把所有图片都处理成了同样宽。

没有 Bootstrap 的布局类，这些图片就从上到下依次排列。如果把窗口宽度缩小到大约 320~400px，可以看到它们垂直排列成一栏时的样子，如下图所示：



在针对大视口调整布局之前，我们先来为说明元素添加样式。

6.7.1 定位及美化说明

我们要把说明元素放到对应用户照片的上面。

(1) 在打开的_page-contents.less 文件中，添加针对`#impact`部分的注释：

```
// Impact Section
#impact {
}
```

(2) 为每个`.hreview`元素添加必要的样式，为下一步定位打下基础。这里添加了相对定位、内边距，去掉了 Bootstrap 缩略图默认的边框：

```
#impact {
   .hreview {
     position: relative;
     padding: 0 10px;
     border: none;
   }
}
```

(3) 现在可以为说明元素添加样式了。我们要在每张图片上添加半透明的背景，并将其绝对定位到图片底部：

```
.hreview {
  ...
  .caption {
    position: absolute;
    top: auto;
    left: 10px;
    right: 10px;
    bottom: 0;
    line-height: 1.1;
    background: hsla(0,0,10%,0.55);
  }
```

(4) 接着，去掉 blockquote 和.reviewer 元素不必要的外边距和内边距，按我们的需要重新设置：

```
#impact .hreview {
    ...
  .caption {
    ...
    blockquote,
    .reviewer {
      margin: 0 6px;
      padding: 0;
  }
```

(5) 下面就是评论文字了，我们要指定外边距、边框、字体、字号和颜色：

```
blockquote {
   margin-top: 4px;
   border: none;
   font-family: @font-family-serif;
   font-size: @font-size-large;
   color: #fff;
}
```

(6) 下面再给评论者的名字指定样式，应该定位到评论内容之下：

```
...
.reviewer {
  margin-top: 2px;
  margin-bottom: 4px;
  text-align: right;
  color: @gray-lighter;
}
```

别忘了把对应的花括号加上。

(7) 保存文件，编译为 CSS，刷新浏览器，应该看到下图所示的结果：



向下滚动，再看看其他评论怎么样。



还不错。不过，我们还能做得更好。

6.7.2　调整说明元素的位置

看看每张图片上的可用空间，再在不同视口宽度下检查一下响应式网格中叠加文本的变化情况。你会发现自己需要针对每个说明元素设置样式，以保证对相应图片位置最合适。

这就是 `review-item-1`、`review-item-2` 这些类可以派上用场的地方。通过它们就可以针对每张图片分别设置样式了。在_page-contents.less 文件中添加如下代码：

```
#impact {
  .review-item-4 .caption {
    top: 0;
    left: 62%;
    right: 10px;
    bottom: auto;
    .reviewer {
      margin-top: 6px;
      text-align: left;
    }
  }
  .review-item-5 .caption {
    top: 0;
    left: 17%;
    right: 10px;
    bottom: auto;
  }
}
```

上面的规则针对特定的评论调整了说明元素的位置，得到了如下结果：



在 06_Code_END 文件夹中的 less/_page-contents.less 文件中，你会发现从第 132 行以后都是我添加的针对每个评论的特殊样式。看完这个结果，没准你会不满意我的调整，位置啊，样式啊，都可能不满意。没关系，你自己可以动手做出自己满意的！

下面我们再考虑图片墙布局。第一步就是指定元素的宽度，这时需要利用 Bootstrap 的网格类。

6.7.3 添加Bootstrap的网格类

利用 Bootstrap 的网格类，可以使用 `col-sm-6` 在小屏幕中实现两栏布局，使用 `col-md-4` 在中大屏幕中实现三栏布局。

每个 `hreview` 元素的类结构都将如下面这行标记所示：

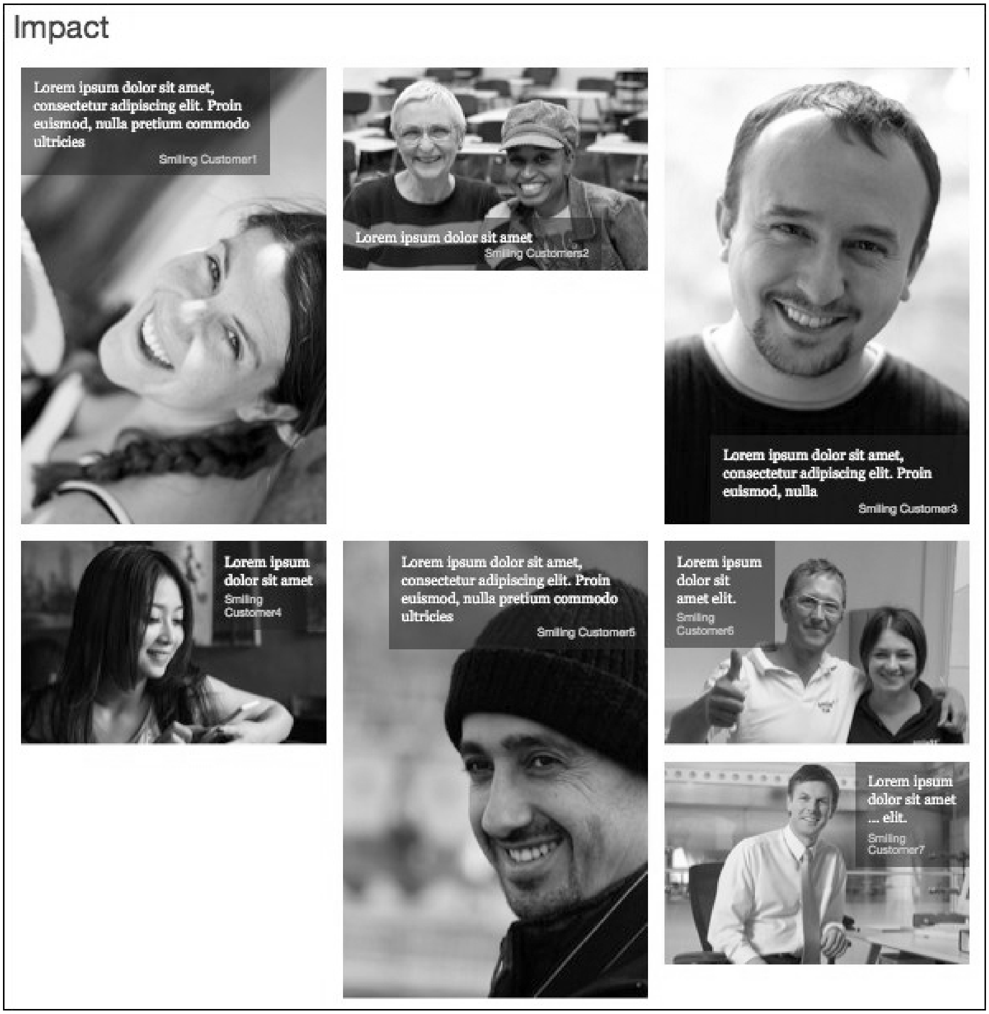
```
<div class="hreview review-item-1 thumbnail col-sm-6 col-md-4">
```

给每个评论都添加这两个类。

保存文件，编译并刷新浏览器。拉伸、收缩浏览器窗口，观察在小视口和中视口中布局的变化情况。可以看到类似下图所示的结果：



在中大视口中，可以看到类似下面的结果：



在前面讨论功能列表的时候，我们说过，如果网格项高度不一，它们就会穿插，不会形成整洁的网格。为此我们给每个元素设定了固定的高度。但在这里，我们希望每个评论的高度不同。既然还需要创建图片墙，那就得借助一点 JavaScript。

6.7.4　下载并链接JavaScript插件

要实现图片墙效果，就得利用 JavaScript 计算可用空间，然后用最合适的图片去填充相应空间，最终让高度不同的块形成整齐的拼贴效果。

为了实现我们想要的效果，可以利用一个叫 Masonry 的 JavaScript 插件，它是由 David DeSandro 开发并维护的。

(1) 在浏览器中打开 http://masonry.desandro.com；
(2) 下载压缩后直接可以使用的 masonry.pkgd.min.js；
(3) 用编辑器打开 masonry.pkgd.min.js，复制其全部内容；
(4) 在项目文件中，打开 js/plugins.js，然后将复制的代码粘贴到 Bootstrap 的 JavaScript 代码后面；
(5) 保存并关闭这个文件。

我们知道，plugins.js 文件已经链接到 index.html 了。因此，以上几步也就把插件的脚本链接到了页面。（虽然这样增大了脚本文件，但却没有增加 HTTP 请求。）

6.7.5 初始化Masonry插件

下面我们要使用 HTML 属性来初始化 Masonry 插件。

> 要了解 Masonry 插件，可以参考它的文档：http://masonry.desandro.com/#getting-started。

在 index.html 文件中，进行如下修改。

(1) 给 `div class="reviews"`添加 `js-masonry` 类，这是所有评论的父元素。这样插件就会知道要在哪里起作用。
(2) 然后，在同一个元素上，添加一个数据属性，指定要拼贴的项。结果标记如下：

```
<div class="reviews js-masonry" data-masonry-options='{"itemSelector":
".hreview" }'>
```

> 注意 `data-masonry-options attribute` 的值使用的单双引号，不要弄错：`data-masonry-options='{ "itemSelector": ".hreview" }'`。

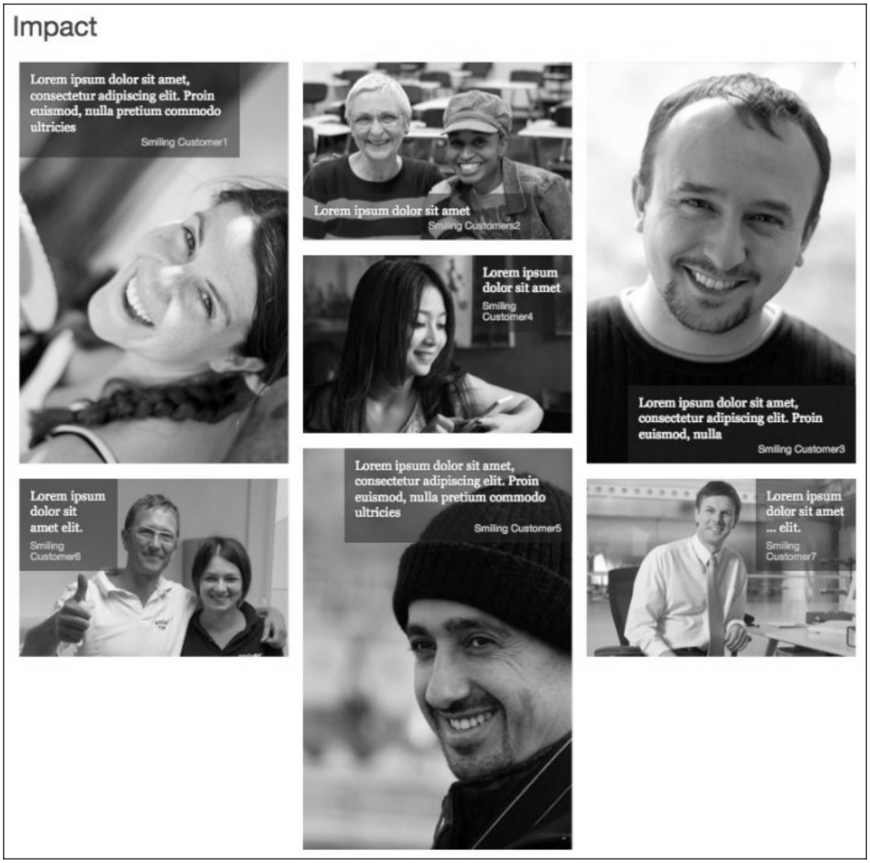
这样就可以告诉插件哪些元素参与拼贴了。这里指定的是 `hreview` 类（当然使用 `thumbnail` 也一样）。

(3) 保存 index.html 并刷新浏览器。你会发现原来存在于图片间的空白一下子就消失了。

小视口中拼贴效果（我们使用 `col-sm-6` 指定了两栏布局）如下图所示：



而下图是在中大视口中的三栏效果：



多拖动浏览器窗口，缩小，放大，经过各个断点试一试。观察一下评论的图片墙在两栏和三栏间切换时的动画效果！

6.7.6 切齐图片

现在离客户要求的结果已经非常近了。可是，我们仍然有一个问题没有解决，就是可能没有准确地检测到图片大小。在两栏的小型布局中，Smiling Customer5 会稍微长出一点来。而在中大型三栏布局中，这张图片伸出得更长了。我们可以提，说把这张图片换掉。但客户说她非常喜欢这一张，不能换。所以我们还得想办法让它在适合整体布局。

好在，我们获得了许可，可以做一些剪裁。换句话说，实在不行，可以不上其他用户的照片。这样我们就可以做点什么了，先从修复三栏布局开始：

(1) 在打开的_page-contents.less 中，添加一行注释：

```
// Cutting and trimming for masonry layout
```

(2) 接下来，我们要在多个断点进行多处调整。但我们不必新写一个媒体查询，而是可以利用 LESS 的嵌套能力，把媒体查询嵌套在#impact 选择符中，在第一个查询中，我们隐藏 Smiling Customer4，如下所示：

```
#impact {
  @media (min-width: @screen-md-min) {
    .review-item-4 {
       display: none;
    }
  }
}
```

(3) 保存文件，编译 CSS，然后刷新浏览器。此时的三栏布局应该完美对齐了，看看下面的屏幕截图吧：



下面接着调整两栏布局。

两栏布局中的 Smiling Customer5 大约只长出了 20px，因此这里只需要把图片从底部切掉一点就行了。

我们只想在小视口中切掉一点图片，不包括超小视口，也不包括中大视口。为此，需要在媒体查询中同时列出最小和最大宽度。在第一个媒体查询后面添加下面这个媒体查询，实现所需效果的代码如下所示：

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) {
    .review-item-5 {
      height: 474px;
      overflow: hidden;
      img {
        width: 100%;
      }
    }
  }
```

这几行代码完成了以下几件事。

- 将 `review-item-5` 的高度精确设置为 474px，以便它与相邻图片底端对齐。
- 对于超高溢出部分，隐藏之。
- 强制图片宽度填满可用空间。

结果非常棒，看看下面的屏幕截图吧：



非常好！

6.7.7　适应小微屏幕

好像 Bootstrap 的响应式网格与图片墙结合起来，在某些浏览器的小微屏幕下会发生冲突。至少我在测试的过程中，发现在小微屏幕里，这些图片不再受控，都显示得非常大。

这是因为 Bootstrap 的 `col-sm-`和 `col-lg-`类对小微屏幕不再适用。因此，`hreview` 缩略图及相应的图片完全不再受约束。

此时，我们有两个选择：

- 给每个评论添加 `col-12` 类；
- 写一点 LESS 添加约束。

如何做取决于你。对于我来说，我选择第二种方式。

为此，我只要在在_page-contents.less 中再添加一个媒体查询：

```
@media (max-width: @screen-xs-max) {
```

在这个媒体查询中，我们来限制 `div class="reviews"`的最大宽度为 400px。这个值既保证图片足够大，也不会让它们太大。我们再使用`.center-block()`混入为评论加入自动的左右外边距，从而实现居中。全部代码如下所示：

```
#impact {
   @media (max-width: @screen-xs-max) {
     .reviews {
        max-width: 400px;
        .center-block();
     }
   }
}
```

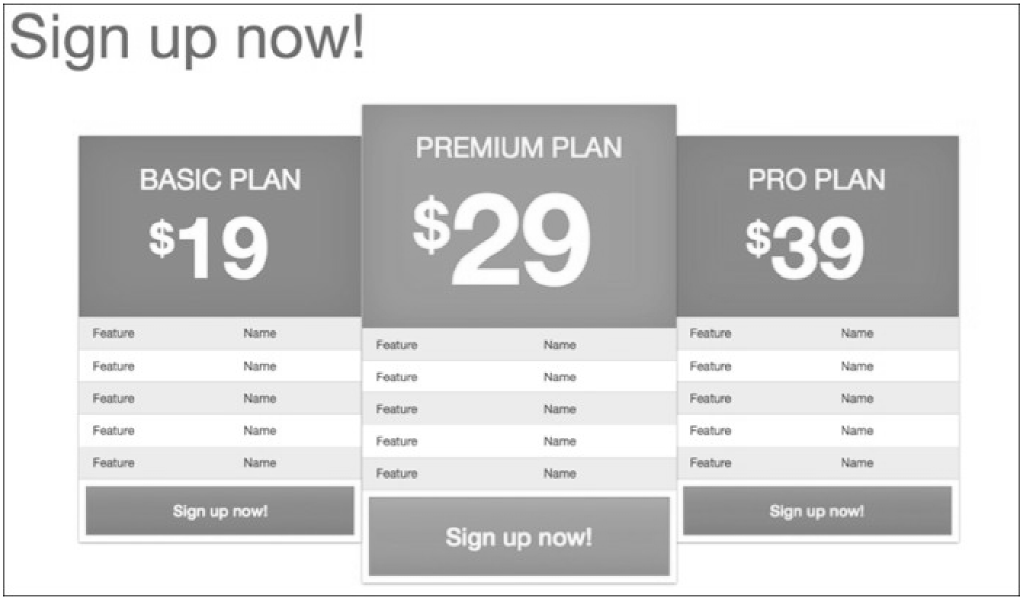
保存文件，编译并刷新浏览器。

好啦，用户评论部分已经完全达到客户要求。

现在我们继续最后一个部分吧。

6.8　吸引人的价目表

我们再来看一眼客户提供的设计图，看看客户期望的结果是什么样的：



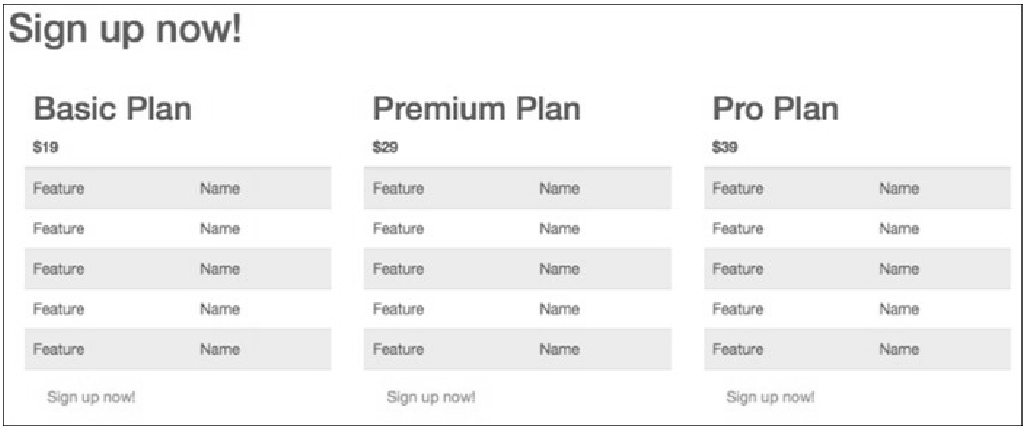
我们得考虑一下要完成这个结果需要做什么，在不同的视口中又需要如何调整它们的布局。

6.8.1 准备变量、文件和标记

如前面的屏幕截图所示，这个设计方案中涉及几个表格。我们可以先从调整与表格相关的几个变量开始。这些变量都在_variables.less 中。搜索表格部分，然后调整与背景、强调的行和边框相关的变量，调整后的结果如下所示：

```
// Tables
// -------------------------
...
@table-bg:      transparent; // overall background-color
@table-bg-accent:  hsla(0,0,1%,.1); // for striping
@table-bg-hover:  hsla(0,0,1%,.2);
@table-bg-active:  @table-bg-hover;
@table-border-color:  #ccc; // table and cell border
```

保存文件，编译为 CSS，然后刷新页面，可以看到下图所示的结果：



这是一个起点。接下来我们需要写更具体的样式。

现在，_page-contents.less 文件越来越长了，而我们眼前的任务则是只关注表格样式。为了保存这些样式，我们再为价目表创建一个 LESS 文件。

(1) 在 less 文件夹中创建_pricing-tables.less。

(2) 在__main.less 中导入_page-contents.less 的后面导入这个文件：

```
@import "_pricing-tables.less";
```

(3) 在编辑器中打开_pricing-tables.less，开始在里面写新样式。

不过，在写新样式之前，我们还是先来看看表格的标记。

> 在每个表格标记的父元素中，我们已经应用了下面的类：
> - package package-basic
> - package package-premium
> - package package-pro

比如，第一个表格，它的父 div 就是这样的：

```
<div class="package package-basic col-md-4">
   <table class="table table-striped">
...
```

类似地，第二和第三个表格的父元素分别加入了 package package-premium 和 package package-pro 类。

这些父容器通过 col-md-4 也提供了基本的布局样式，即在中型视口中会排成三栏。

下面我们分析看一看每个表格的标记。在第一个基本配置表中，已经应用了 table-striped 类：

```
<table class="table table-striped">
```

这个表格使用<thead>元素作为最顶层的包含块。在这个元素内部，是一个跨两列的<th>，其中包含<h2>标题，是配置名称，还有一个<div class="price">，里面是价格：

```
<thead>
  <tr>
    <th colspan="2">
      <h2>Basic Plan</h2>
      <div class="price">$19</div>
    </th>
  </tr>
</thead>
```

再后面是包含 Sign up Now!按钮的 tfoot 标签：

```
<tfoot>
  <tr><td colspan="2"><a href="#" class="btn">Sign up
    now!</a></td></tr>
</tfoot>
```

然后是 tbody 标签，包含一组功能列表，很直观，每行两列：

```
<tbody>
  <tr><td>Feature</td><td>Name</td></tr>
  <tr><td>Feature</td><td>Name</td></tr>
  <tr><td>Feature</td><td>Name</td></tr>
  <tr><td>Feature</td><td>Name</td></tr>
  <tr><td>Feature</td><td>Name</td></tr>
</tbody>
```

最后，当然是两个关闭标签：

```
  </table>
</div><!-- /.package .package-basic -->
```

其他两个表格的结构也都一样。

这就是我们下一步工作的基础。

6.8.2 表格头

要美化所有表格的表格头元素，需要做以下几件事：

- 居中文本；
- 添加与最终版本接近的中性灰作为背景颜色；
- 把字体颜色改为白色；
- 把 h2 转换为大写；
- 增大价目表的尺寸；
- 给表格添加必要的内边距。

6

完成以上美化工作，只要下面几行代码即可。这里我们把所有针对表格的样式都放到 #signup 选择符中：

```
#signup {
  table {
    border: 1px solid @table-border-color;
    thead th {
      text-align: center;
      background-color: @gray-light;
      color: #fff;
      padding-top: 12px;
      padding-bottom: 32px;
      h2 {
        text-transform: uppercase;
      }
    }
  }
}
```

简单来说，这些样式完成了除增大价目表尺寸之外的所有工作。我们可以在这个基础上，开始添加样式，仍然在#signup 选择符内：

```
.price {
      font-size: 7em;
      line-height: 1;
}
```

这样就得到了下面的结果：



这就跟我们预期的结果接近了，但我们想减少美元符号的大小。为了能控制到它，必须在标记中给它加个 span 标签：

```
<em class="price"><span>$</span>19</em>
```

别忘了其他两个表格也要如法炮制。

添加新标签后，可以把相应规则嵌套在.price 中：

```
.price {
    ...
    span {
        font-size: .5em;
        vertical-align: super;
    }
```

以上规则就缩小了美元符号为原来的一半，并且顶部对齐。

接下来居中结果，需要给父.price 选择符添加一点负外边距：

```
.price {
  margin-left: -0.25em;
  ...
```

下面的屏幕截图就是现在的结果：



6.8.3 表体和表脚

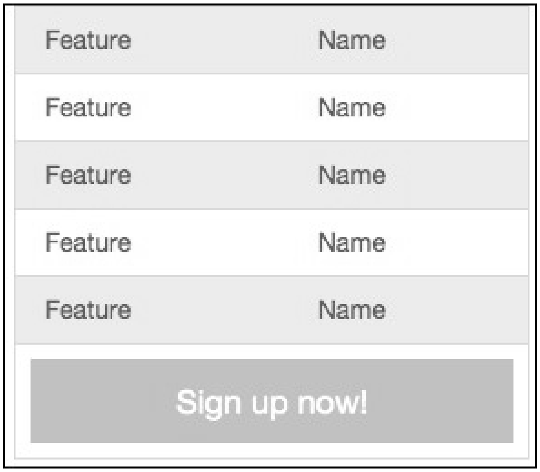
同样以三个价目表为目标，统一作如下调整：

- 给功能列表添加左、右内边距；
- 把按钮拉伸至全宽；
- 增大按钮尺寸。

用下面的规则就可以：

```
#signup {
  table {
    ...
    tbody {
      td {
        padding-left: 16px;
        padding-right: 16px;
      }
    }
    a.btn {
      .btn-lg;
      display: block;
      width: 100%;
      background-color: @gray-light;
      color: #fff;
    }
  }
}
```

保存文件，编译到 CSS，刷新浏览器。应该可以看到下面的结果：



公共的样式完成了，接下来可以考虑差异化了。

6.8.4 为不同的价目表添加不同的样式

我们先来给不同的价目表的表头和 Sign up now!按钮添加预期的颜色。在客户给我们的设计图中，Basic 是蓝色，Premium 是绿色，Pro 是红色。下面我们将选择好的颜色值指定给三级品牌色，如下所示：

```
@brand-primary:         #428bca;
@brand-secondary:       #5cb85c;
@brand-tertiary:        #d9534f;
```

设置完颜色变量，就可以将它们应用给适当的表头和按钮了。此时要用到前面给每个表格的父元素添加的特定的类，也就是 `package-basic`、`package-premium` 和 `package-pro`：

(1) 在 less/_pricing-tables.less 中，新写一段注释：

```
// Pricing Table Colors
```

(2) 在这里我们给 `.package-basic` 表应用主品牌色 `@brand-primary`，先在 `thead th` 元素中试验一下：

```
#signup .package-basic table {
  thead th {
    background-color: @brand-primary;
  }
```

(3) 然后再把主品牌色应用给 `thead th` 元素的按钮。这里，我们还使用了 bootstrap/mixins.less 中定义的 `.button-variant()` 混入给 `:hover` 和 `:active` 状态应用样式。这个混入函数接受三个参数：颜色、背景颜色和边框颜色。我们是这样写的：

```
  ...
  .btn {
    .button-variant(#fff; @brand-primary; darken(@brand-primary, 5%));
  }
}
```

(4) 编译之后，这个简洁的混入函数就会给按钮及其悬停、活动状态生成对应的样式！

> 要了解.button-variant()的原理，可以参考 bootstrap/mixins.less 中的定义，以及 bootstrap/buttons.less 文件，其中使用这个混入函数定义了 Bootstrap 默认的按钮类。

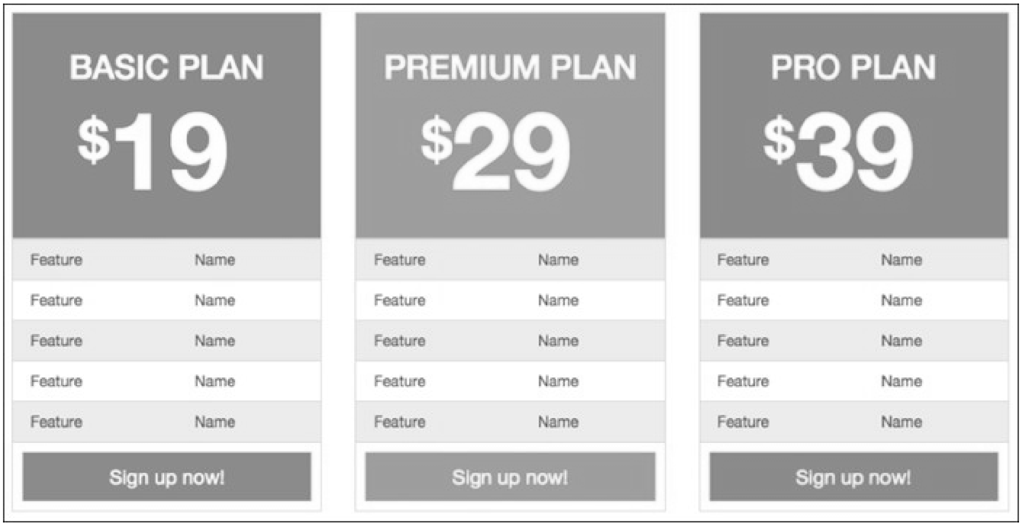
(5) 现在，需要对.package-premium 表重复上述过程，只不过这次要使用@brand-secondary 变量：

```
#signup .package-premium table {
  thead th {
    background-color: @brand-secondary;
  }
  .btn {
    .button-variant(#fff; @brand-secondary; darken(@brand-secondary, 5%));
  }
}
```

(6) 最后，再给.package-pro 表应用第三品牌色@brand-tertiary：

```
#signup .package-pro table {
  thead th {
    background-color: @brand-tertiary;
  }
  .btn {
    .button-variant(#fff; @brand-tertiary; darken(@brand-tertiary, 5%));
  }
}
```

(7) 保存文件，编译为 CSS，刷新浏览器。应该看到应用了新颜色的表格，如下图所示：

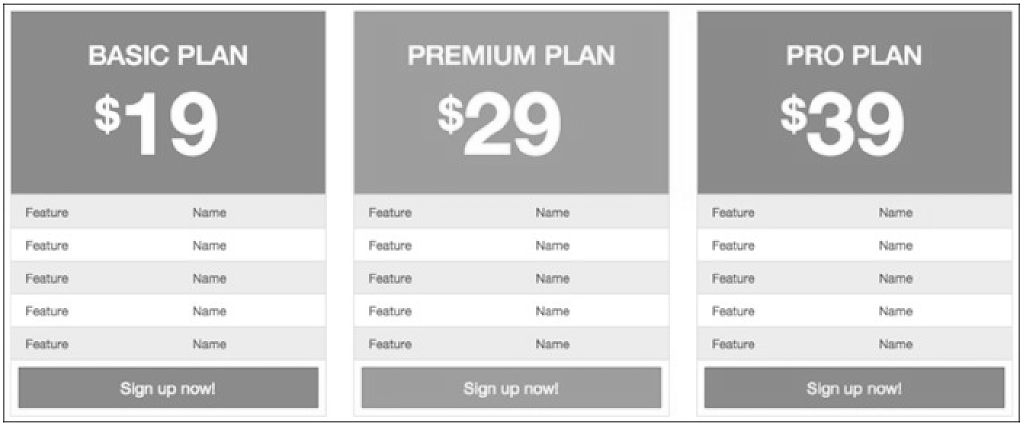


非常好！

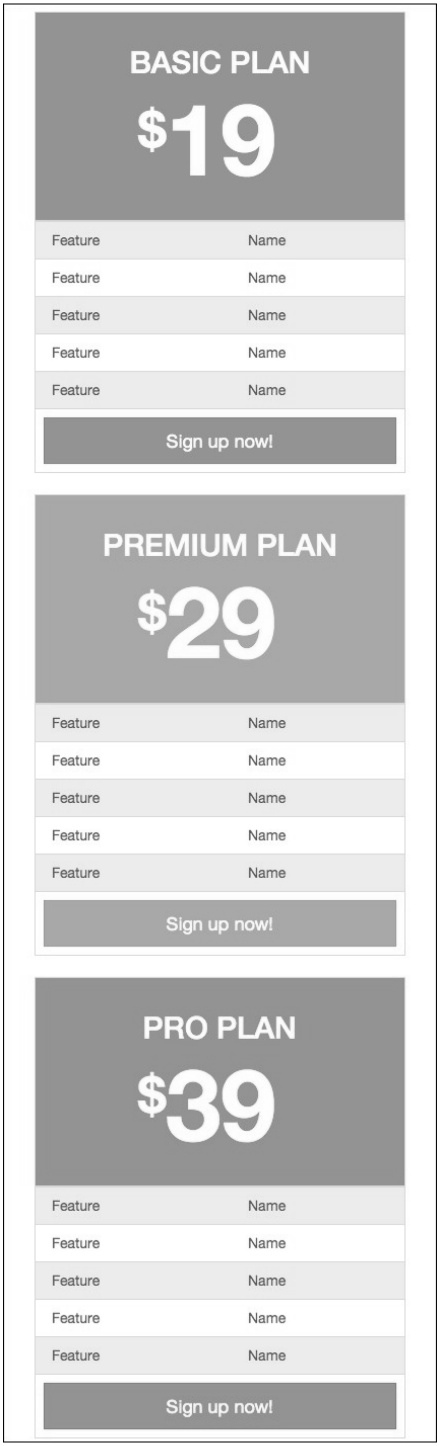
好了，接下来解决不同视口适配的问题。

6.8.5　适配小视口

由于 Bootstrap 3 对响应式设计的重视，我们的表格在视口断点切换时都表现得很好。前面已经看到在中级宽度视口中表格的表现了，下面再看看在视口更宽时表格的布局：



在窄视口中，表格上下堆叠起来，如下图所示，也非常漂亮：


BASIC PLAN
$19
Feature Name
Feature Name
Feature Name
Feature Name
Feature Name
Sign up now!
PREMIUM PLAN
$29
Feature Name
Feature Name
Feature Name
Feature Name
Feature Name
Sign up now!
PRO PLAN
$39
Feature Name
Feature Name
Feature Name
Feature Name
Feature Name
Sign up now!

可是，在大约 480~992px 之间的时候，表格会扩展到与屏幕一样宽。很明显，这时候就太宽了，如下图所示：



因为只有三个表格，所以不可能考虑两栏布局的方案。只能限制表格宽度，并使用自动的左、右外边距使它们居中。我们使用 max-width 为@screen-sm-max 的媒体查询，把表格的最大宽度设置为 400px，再使用.center-block()让表格居中：

```
//
// Constrain width for small screens and under
// ------------------------------------------

@media (max-width: @screen-sm-max) {
  #signup .package {
```

```
        max-width: 400px;
        .center-block();
    }
}
```

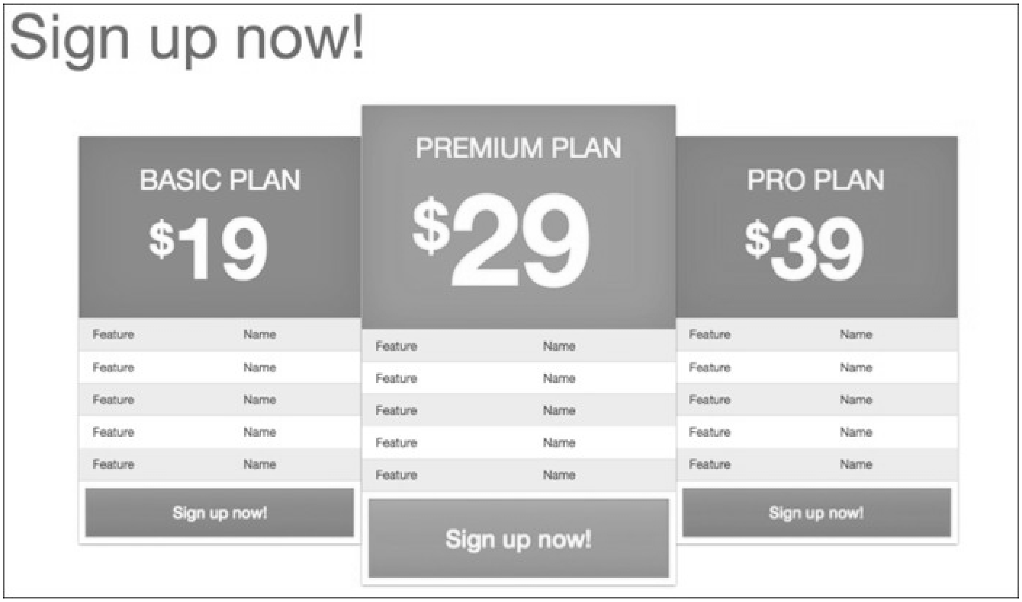
保存文件，编译为 CSS，然后刷新浏览器。应该可以看到宽度受限的表格在窗口内居中了，如下图所示：



此时，三个表格有了差异，而且具备了响应性。可是，还差一点呢。在中、大视口，我们希望 Premium 方案能够更突出。

6.8.6 突出重要的表格

再看一眼设计图，就会发现我们的设计应该达到目标：至少在桌面级视口中，中间档价目表的文字应该更大，而且从视觉上应该在另外两个表格的上方，如下图所示。



这个效果通过调整内边距、外边距和字号就可以实现。

我们要在针对中大视口的媒体查询中添加样式：

```
//
// Visually enhance the premium plan
// ------------------------------------------
@media (min-width: @screen-md-min) {
}
```

在这个媒体查询中，我们首先减少 Basic 和 Pro 表（即第一和第三个表）的宽度，再给它们添加一些上外边距，将它们向下推一下：

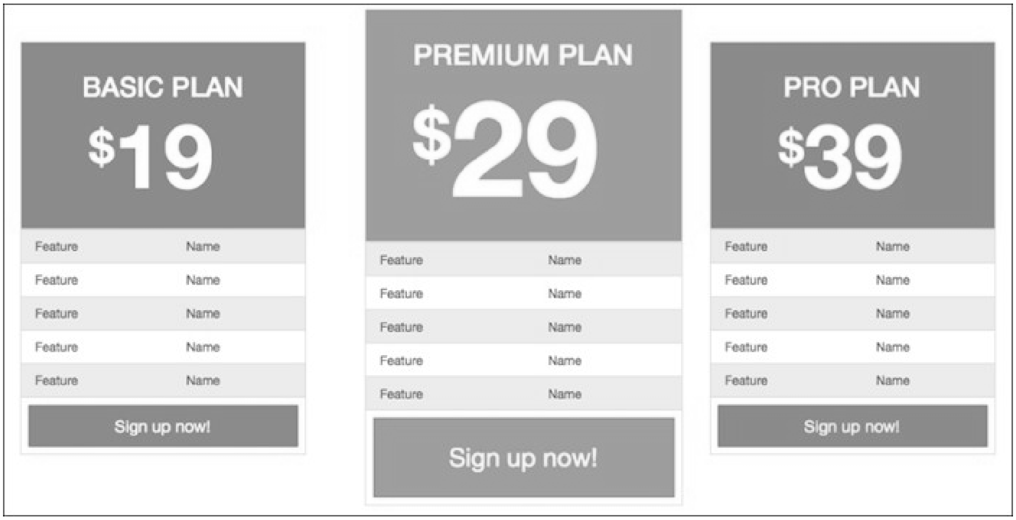
```
// Size down the basic and pro
#signup .package-basic table,
#signup .package-pro table {
  width: 90%;
  margin-top: 36px;
}
```

接下来增大 Premium 表的字号，并为其按钮添加内边距：

```
// Size up the premium
#signup .package-premium table {
  thead th {
    font-size: 1.5em;
    h2 {
      font-size: 1.5em;
    }
  }
```

```
  a.btn {
    font-size: 2em;
    padding-top: 24px;
    padding-bottom: 24px;
  }
}
```

这样得到的结果跟预期目标已经接近了，如下图所示：



下一个目标就是让三个表格靠近一些。为此，就要对外边距进行一些调整，再用一用 z-index 属性：

```
// Squeeze tables together
#signup .package-basic {
  margin-right: -58px;
  margin-left: 58px;
  z-index: 1;
}
#signup .package-premium {
  z-index: 1000;
}
#signup .package-pro {
  margin-left: -30px;
  z-index: 1;
}
```

解释一下上面的规则吧。

- 使用负的右外边距把（左侧的）BASIC PLAN 表向右推，同时用等量的左外边距抵消它，以保持三个表格的相对位置不变。
- 使用负的左外边距把（右侧的）PRO PLAN 表向左推。

6

❑ 调整所有表格的 z-index 值，让左、右两个表位于中间的表底下。



> 关于 z-index，可以参考这篇文章：http://css-tricks.com/almanac/properties/z/z-index/。

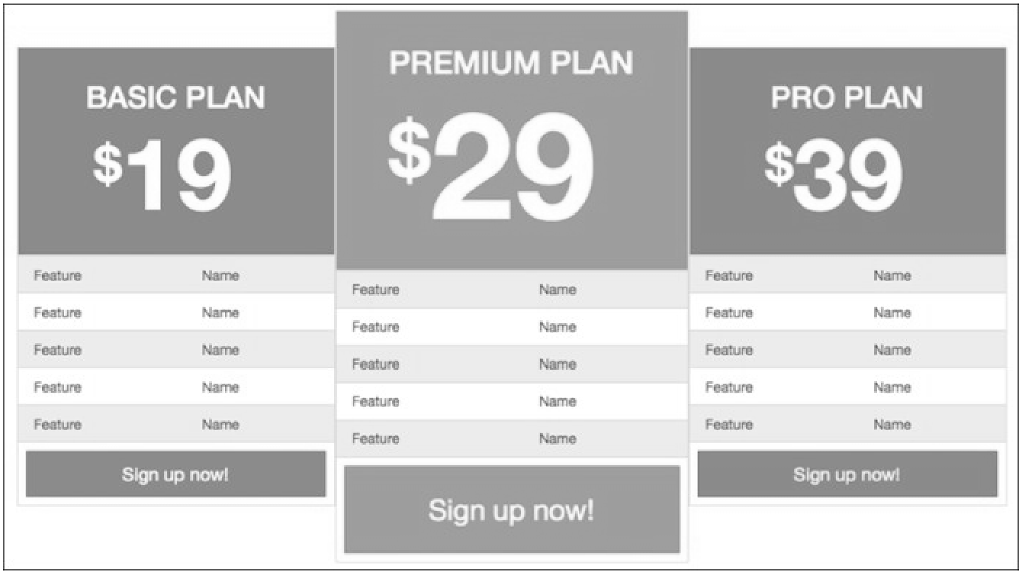
下面的屏幕截图显示了在中等宽度视口中的效果：



快要完工了。接下来只需要再对 Basic 表在下一个更大的断点作一调整。在上一个媒体查询后面再写一个新的媒体查询：

```
@media (min-width: @screen-lg-min) {
  #signup .package-basic {
    margin-right: -65px;
    margin-left: 65px;
  }
}
```

保存文件，编译到 CSS，刷新浏览器。应该看到 1200px 及更大的视口中的效果如下：



就这样了！至此，我们完成了客户给我们提出的最后一个要求。

现在，从整体角度做一些修饰和调整的工作。

6.9 最后的调整

本节，我们将从增强页面整体性的角度出发，再做一些细节的调整。首先，给页面中的每个部分的 `h1` 标题增加必要的上、下内边距，并增大字号。然后，再增强一下导航的体验，即给导航条添加 ScrollSpy 并使用 jQuery 将点击导航后的滚动过程变成动画。

先来增强各部分的主标题。现在看一下这些标题，你会发现它们很不起眼。比如，就以 Features 部分为例吧：



我们的增强方案是降低其对比度，增大其内边距。我们只想把规则应用给 FEATURES、IMPACT 和 SIGN UP，因此可以通过 ID 选择它们。

(1) 在编辑器中打开_page-contents.less。

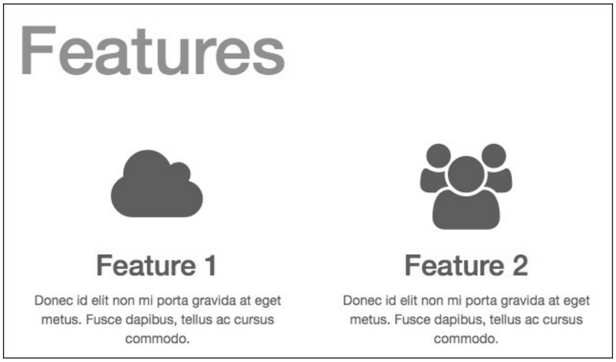
(2) 在文件顶部，在给页面主体应用上内边距的规则之后，添加以下代码：

```
#features, #impact, #signup {
  padding-top: 36px;
  padding-bottom: 48px;
  h1 {
    font-size: 5em;
    color: @gray;
    line-height: 1.3;
    padding-bottom: 24px;
    }
}
```

(3) 以上规则做的事情如下：

- 给这些部分添加上、下内边距；
- 显著增大 h1 标题的字号；
- 减少标题的对比度；
- 通过设置行高和下内边距，保证标题周围的空间合适。

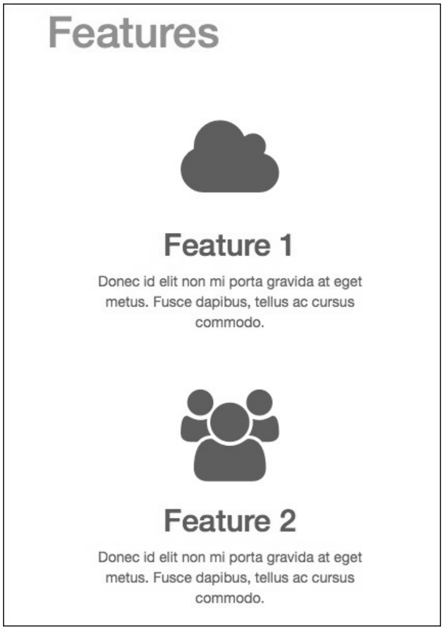
(4) 保存，编译，刷新，看看有什么不一样：



这些变化会体现在所有视口大小的页面中。对于小视口，目前的 h1 太大了。另外，我们还需要添加一些左、右外边距。因此还要继续调整一下。我们不想让后面的样式影响大视口下的布局，所以得把它们封装到一个媒体查询中：

```
// Adjust section headings for extra-small viewports only
@media (max-width: @screen-xs-max) {
  #features, #impact, #signup {
    margin-left: 30px;
    margin-right: 30px;
    h1 {
      font-size: 3em;
    }
  }
}
```

下面的屏幕截图展示了调整后的效果：



改进很大。

接下来我们改进导航的体验。

6.10　为导航条添加 ScrollSpy

我们要配置顶部的导航条，令其对应页面中的位置。下面给导航条添加 Bootstrap 的 ScrollSpy：

> 这里是 Bootstrap ScrollSpy 插件的文档：http://getbootstrap.com/javascript/#scrollspy。

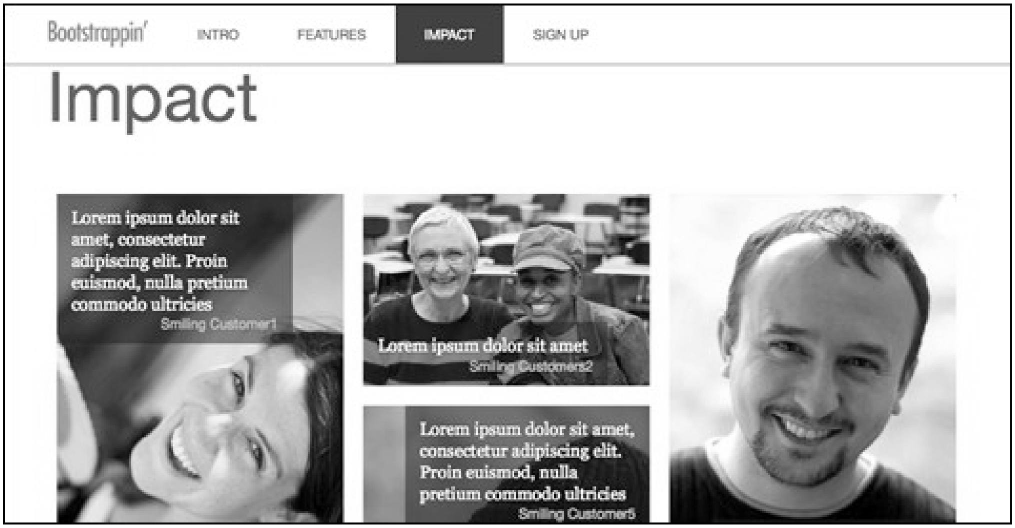
(1) 在编辑器中打开 index.html。

(2) 给 `body` 标签添加下面的 ScrollSpy 属性：

```
<body data-spy="scroll" data-target=".navbar">
```

> 假如页面中包含多个导航条，需要在 `data-target` 属性中具体指出。或许得为 ScrollSpy 导航条添加一个 ID，比如 `id="navbar-primary"`，然后将这个 ID 作为 `data-target` 属性的值。

(3) 设置了这些属性后，保存文件，刷新浏览器，点击导航，会发现主导航能够定位到页面中对应的位置，如下图所示：



添加动画

下面给点击导航后的页面滚动添加动画，为此需要在 main.js 文件中添加几行代码。

(1) 在编辑器中打开 js/main.js。

(2) 在`$(document).ready(function() {`中添加以下代码：

```
$('#navbar [href^=#]').click(function (e) {
  e.preventDefault();
  var div = $(this).attr('href');
  $("html, body").animate({
    scrollTop: $(div).position().top
  }, "slow");
});
```

(3) 保存并刷新浏览器。

刚才的代码做了什么？我们使用 jQuery 做了以下几件事。

- 选择了`.navbar` 元素中以页面位置中的锚为目标的链接；
- 阻止了默认的单击行为；
- 将滚动过程变成动画，设置了动画速度为 `slow`。

单击某个导航项，应该可以看到滚动动画了！

6.11 小结

花点时间前后翻阅一下页面，欣赏一下各个部分的细节，调整一下窗口，看看布局的响应性如何。

想一想，一个页面就实现了那么多功能，而且它能够适配桌面浏览器、平板浏览器和手机浏览器，应该有不小的成就感吧！

下面来回顾一下，我们的客户向我们提出了设计一个单页营销站点的要求：

- 使用 Bootstrap 高清图样式的大字欢迎语，背景图片十分抢眼，而且具有响应能力；
- 使用 Font Awesome 图标的功能列表；
- 图片墙网格的用户赞誉，同样完美适配各种视口；
- 注册区使用 Bootstrap 的表格样式，并自定义了中档价目表，其在中、大视口中更加突出；
- 使用 ScrollSpy 和 jQuery 增强了导航条，并添加了动画滚动效果。

实现了上述设计之后，应该说，没有什么是我们不能通过 Bootstrap 实现的了。

做完本章和前面几章的项目，相信你一定有了很大收获。总结一下吧：

- 掌握了 Bootstrap 的所有细节；
- 把 Bootstrap LESS 和 JavaScript 整合进我们的项目文件；
- 把 Bootstrap 的 glyphicons 替换成了更丰富的 Font Awesome 图标字体；
- 对 Bootstrap 的样式进行自定义和调整，从而达到对设计结果的精确控制。

不要忘了，在本书的附录部分，还有针对所有项目的指南。比如，针对上线部署优化 Bootstrap 的资源（附录 A），利用当前最新技术实现响应式图片（附录 B），给传送带添加手势（附录 C）。

除此之外，还有很多优秀的资源可供我们进一步探索 Bootstrap。其中，Bootstrap 社区就是一个非常活跃和值得关注的地方。Bootstrap 无疑是 Web 前端发展史上的一座里程碑，值得我们学习。

优化站点资源

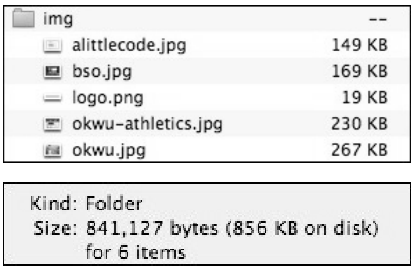
A

速度很重要。用户很关心。我们的站点必须加载够快，否则用户就会走人。SEO 也很重要。我们的站点必须加载够快，否则搜索排名就会下降。

明白了这些，我们就来清点一下第 2 章人个作品站点中的资源。特别地，来看一看我们能控制的、影响页面速度的重要因素——文件大小，包括图片、CSS 和 JavaScript 文件。只要简单几步，我们就可以给这些文件“瘦身”，缩短加载时间。

A.1 优化图片

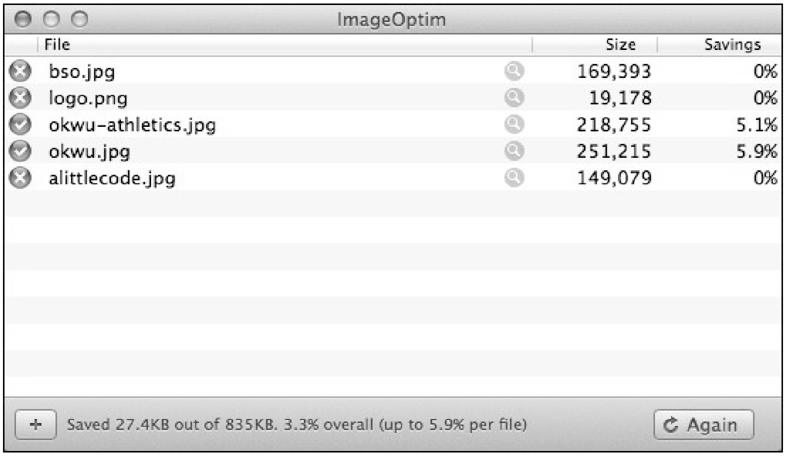
这些图片都通过 Photoshop 的“保存为 Web 格式”进行了一定程度的优化。但是，所有图片加在一块，也有 856 KB。



这些图片很重要。（毕竟是个人作品站点啊。）可是，这个体量也确实大了一些。附录 B 还会向大家介绍一种响应式图片技术，可以进一步减少小屏幕设备中的文件大小。不过即使不用该技术，通过更有效的压缩，照样还能减少文件大小。

要减少文件大小，同时又不会损害图片质量，可以使用一些工具，比如 Yahoo!的 Smushit：http://www.smushit.com/。

对于 Mac 用户，免费的 ImageOptim 应用（http://imageoptim.com/）也能达到类似的目的。使用该应用，可以把整体大小减少 29 KB。



这个成果并不十分明显，但能小些总是好的。

A.2　优化CSS

先看看未优化的样式表 main.css 的文件多大：



137 KB！任何负责任的开发者都不会让这么一个小网站带那么大的样式表。

好消息是，我们可以轻易把这个大小减半。利用 Bootstrap 的模块化 LESS 方案，可以立即缩小 CSS，步骤如下。

(1) 打开 less/__main.less；

(2) 注释掉不需要的 LESS 文件，比如这些：

```
// @import "bootstrap/glyphicons.less";
...
// @import "bootstrap/dropdowns.less";
// @import "bootstrap/button-groups.less";
// @import "bootstrap/input-groups.less";
...
// @import "bootstrap/breadcrumbs.less";
// @import "bootstrap/pagination.less";
// @import "bootstrap/pager.less";
// @import "bootstrap/labels.less";
// @import "bootstrap/badges.less";
// @import "bootstrap/jumbotron.less";
// @import "bootstrap/thumbnails.less";
// @import "bootstrap/alerts.less";
```

```
// @import "bootstrap/progress-bars.less";
// @import "bootstrap/media.less";
// @import "bootstrap/list-group.less";
// @import "bootstrap/panels.less";
// @import "bootstrap/wells.less";
// @import "bootstrap/close.less";
...
// @import "bootstrap/modals.less";
// @import "bootstrap/tooltip.less";
// @import "bootstrap/popovers.less";
```

(3) 当然得小心一点，否则一不留神就可能注释掉必要的文件。因此事后要花点时间重编译，全面测试一下。
(4) 注释掉不必要的文件后，选中编译器中的最小化（或者压缩输出）选项，最后重编译一遍，保存为 css/main.css。
(5) 再看看文件有多大。我这里的结果是只剩下 74 KB，相当于原先的 62%。

当然，你还可以优化得再细一些。比如，可以打开每个保留的 LESS 文件，再把其中没有必要的代码一行一行注释掉。这些练习就留给你们啦！

最后，我们来看看如何优化 JavaScript。

A.3　优化JavaScript

为优化 JavaScript，我们要把 plugins.js 文件中的 Bootstrap 插件，替换成只剩我们用到的几个。然后再重新压缩文件。

(1) 打开 js/plugins.js。
(2) 删除属于 bootstrap.min.js 的代码块。
(3) 打开 js/bootstrap 文件夹，这里面保存着 Bootstrap 插件的独立文件。逐个打开下列文件，将它们的代码复制到 plugins.js 文件里，这三个插件是我们网站中用到的：

- carousel.js
- collapse.js
- transition.js

(4) 保存“瘦身”版的 plugins.js 文件，刷新浏览器进行测试。

- 确保响应式导航条在窄视口中能够折叠，并且单击按钮可以展开下拉列表；
- 确保传送带一切如常。

如果都没有问题，说明已经包含了所需的 JavaScript。

(5) 下一步可以缩小（minify）或“丑化”（uglify）plugins.js 文件了。建议使用下列在线工具。

- ❑ UglifyJS：http://marijnhaverbeke.nl/uglifyjs
- ❑ YUI Compressor：http://refresh-sf.com/yui/
- ❑ 谷歌的 Closure Compiler：http://closure-compiler.appspot.com/

打开这些在线工具，把 plugins.js 的代码复制过去，运行，再把得到的代码复制回 plugins.js。

我们在这里选用 CodeKit，是 Mac 的一款付费应用：http://incident57.com/codekit/。

(6) 保存压缩后的文件。
(7) 比较文件大小。

为对比方便，我为所有文件保存了备份：

- ❑ plugins-all.js 包含完整的 bootstrap.min.js 代码；
- ❑ plugins-uncompressed.js 包含我们需要的三个插件，未压缩；
- ❑ plugins.js 是最终文件，缩小并去空格串联的版本。

下图展示了这几个文件大小的对比：



最终文件只相当于原来的三分之一！

A.4　优化结果

总体来看，我们的优化工作取得了成效。把图片、CSS 和 JavaScript 都算一块，原来的大小是 1021 KB。

优化之后，变成了 909 KB，节省了 112 KB，超过了 10%。

而如果单看 CSS 和 JavaScript，则节省幅度超过了 50%。这个差别就很大了，其效果迟早会显现。

事实上，我们还能够继续优化，尤其是针对小屏设备，方法就是实现响应式图片。关于响应式图片，我们会留在附录 B 里讨论。

实现响应式图片 B

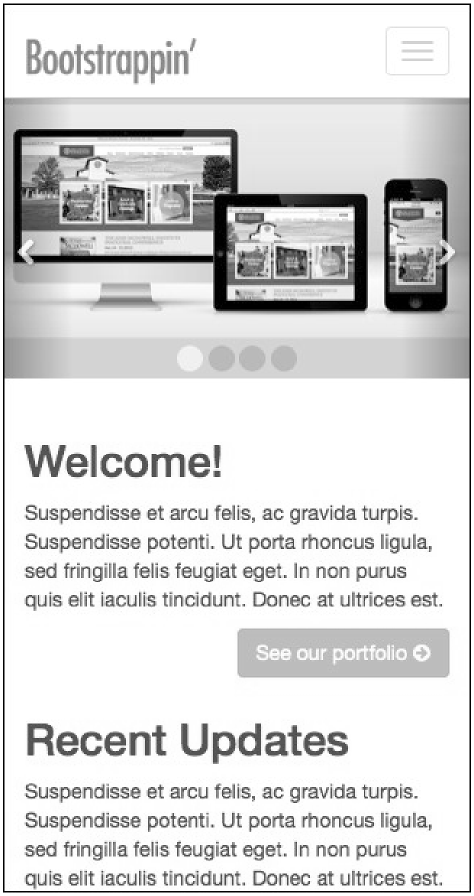
如果我们秉承移动友好的开发宗旨，那么就需要选择一种响应式图片技术。在这个附录中，我们会基于目前比较前沿的技术，来提升第 2 章个人作品站点中作品图片传送带的性能和设计。

B.1 分析作品传送带

在第 2 章的个人作品网站中，传送带中的图片是为全宽布局设计的，宽度是 1600px，大小为 135~189 KB。把这么大的图片发送到手机和非视网膜屏的平板就过分了。在移动优先响应式设计的时代，这样做又是不负责任的。

而且，如果再看一看小屏幕中的显示效果，你可能会发现传送带中的图片应该更高点、窄点才好，因为窄屏幕垂直方向上空间相对富余一些。

在手机屏幕那么宽的视口中，我们的图片，为大屏幕准备的图片，是可以显示，但如果能够更多利用垂直空间，效果会更好。这一点通过下面的屏幕截图可以看出来：



好的响应式图片技术，应该能让我们为小屏幕提供适当的图片，满足小文件、快速加载，以及改进设计的要求。

B.2　选择方案

响应式图片技术的标准还在制定过程中。但目前还没有哪家的方案被采用，浏览器也没有都实现哪一种手段。为此，当前最好的技术要么是服务器端技术，要么是客户端技术。

Smashing 杂志发表过一篇不错的文章叫 Choosing a Responsive Image Solution，作者 Sherri Alexander。这篇文章介绍了前沿的几种技术方案，大家有时间可以看一看，地址是：http://mobile.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/。

在这篇文章乃至其他文章中，Scott Jehl 的 Picturefill 技术都被推崇为一个不错的方案。这是因为 Picturefill 方案较好地平衡了性能和设计问题，而且方案也相当简明。

Picturefill 实现响应式图片只需简单几步：

(1) 准备好针对目标视口的理想图片；
(2) 下载并包含 Picturefill 的 JavaScript 文件；
(3) 用 Picturefill 的标记模式来引入图片。

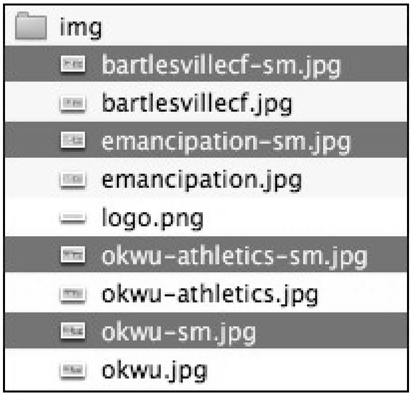
一如往常，实际开发过程还有两个步骤：

- 测试；
- 按需调整。

下面我们就一步一步来做。

B.2.1　准备响应式图片

打开本附录的练习文件，你会发现 img 文件夹中包含了一些特殊尺寸和经过优化的图片。特别是其中一些带-sm.jpg 后缀的图片：



打开这个图片，会发现它们更窄一些，长宽比更小。这是为了像下面这样在窄视口中更多利用垂直空间：



当然，图片也小一些，900px × 600px，保证在视网膜屏中也能有足够的像素，但比起最初的 1600px × 800px 就小多了。这些小图片平均都比原来的大图片小 50%以上。

具体的大小可以参考下面这张截图，注意以–sm.jpg 结尾的小图：

bartlesvillecf-sm.jpg	71 KB
bartlesvillecf.jpg	135 KB
emancipation-sm.jpg	78 KB
emancipation.jpg	156 KB
logo.png	19 KB
okwu-athletics-sm.jpg	94 KB
okwu-athletics.jpg	189 KB
okwu-sm.jpg	93 KB
okwu.jpg	188 KB

图片准备好以后，接下来该 JavaScript 上场了。

B.2.2　使用JavaScript

Picturefill 的文件及文档位于 GitHub，地址是：https://github.com/scottjehl/picturefill。

大家可以花点时间看看文档。我们一会就要用到文档中推荐的元素。但现在，我们要下载 JavaScript 文件，然后用到我们的项目中。好，按照下面的步骤操作。

(1) 在 Picturefill GitHub 的代码库中，找到并下载 picturefill.js。

(2) 复制文件中的代码，包括开头的注释。我们要把它复制到 plugins.js 文件中。

(3) 打开 js/plugins.js，把 Picturefill 的代码贴在其他插件之前或之后，都可以。

(4) 保存。（当然，在部署的产品环境之前，别忘了最小化和压缩 plugins.js。）

接下来，按照 Picturefill 的约定准备标记。

B.2.3　修改标记结构

在 index.html 中，修改每张图片的标记，使用 Picturefill 嵌套 span 元素的模式。这里的标记默认采用小图片，但视口在 640px 及以上的浏览器和 IE8 例外。

以下就是修改后第一张图片的标记。

```
<span data-picture data-alt="OKWU Homepage">
  <span data-src="img/okwu-sm.jpg"></span>
  <span data-src="img/okwu.jpg" data-media="(min-width: 640px)"></span>
  <!--[if (lt IE 9) & (!IEMobile)]>
    <span data-src="img/okwu.jpg"></span>
  <![endif]-->
  <noscript>
    <img src="img/okwu.jpg" alt="OKWU Athletics Homepage">
  </noscript>
</span>
```

下面简单解释一下以上代码。

- 最外层的 span 标签使用 data-picture 属性将整个元素标记为响应式图片。
- data-alt 属性中包含图片的替代文本。
- 开头的<span data-src ...>标签指定了小设备使用的默认图片。
- data-media 属性用于指定在什么条件下使用大图片。在这里，我们指定的是 min-width: 640px，也就是只有视口宽度超过 640px 时才会加载大图片。

> 正如文档中所说，这里的条件还可以使用设备像素比来针对高密度或视网膜屏幕。

- 接下来的条件注释可以确保不支持媒体查询的 IE8 加载大图片。

- 最后的 no-script 元素中包含一个标准的 img 标签，只会在浏览器不支持或禁用 JavaScript 时显示。

要了解更多，请大家参考这里的文档：https://github.com/scottjehl/picturefill。

接下来，请大家依照第一张图片，为其他图片也添加类似的标记。

B.2.4　测试与调整

保存并测试，你会发现这一次传送带的图片不会调整适应屏幕宽度了。这是因为 Picturefil 的标记没有使用 Bootstrap 传送带样式中的选择符。

我们得修改_carousel.less 文件中相应的选择符，好让图片撑满可用空间，步骤如下。

(1) 打开_carousel.less。

(2) 搜索到下面的代码，把> img 和> a > img 子选择符，替换成简单的后代 img 选择符，以便选中现在在 Picturefill 标记中嵌套较深的图片：

```
// Account for jankitude on images
// > img, // commented out
// > a > img // commented out
img { // added to apply to PictureFill responsive image solution
  .img-responsive();
  line-height: 1;
  min-width: 100%; // added
  height: auto; // added
}
```

问题就这样解决了！

B.3　最终的结果

在视口小于 640px 时，传送带应该使用较小但相对较高的图片。



以此为起点，大家还可以参考 Picturefil 的文档，根据需要再调整和适配自己需要的版本。

让传送带支持手势

C

在触摸屏设备中，支持手势轻扫来切换传送带图片是一个非常实用的功能。本附录将为 Bootstrap 的传送带添加轻扫手势支持。

C.1 有什么选择

目前，还没有跨设备测试触摸手势的简便方法。现有条件下的最佳实践，就是要保证添加的手势实用，而且不会影响标准的鼠标事件。其实，如果要让传送带支持轻扫手势，只要一个 JavaScript 插件和几行代码就行。

Justin Lazanowski 专门为实现 Bootstrap 3 传送带的手势交互写过一篇文章，提到三种选择。他的文章在这里，大家可以自己看一看：http://lazcreative.com/blog/adding-swipe-support-to-bootstrap-carousel-3-0/。

本附录将使用 jQuery 插件 TouchSwipe，GitHub 地址为：https://github.com/mattbryson/TouchSwipe-Jquery-Plugin。

使用这个插件，可以通过下列步骤让传送带支持轻扫手势：

(1) 把 TouchSwipe 插件包含到我们的插件文件中；
(2) 在 main.js 文件中写几行调用代码。

很简单，下面我们来做。

C.2 取得并包含TouchSwipe插件

按照下面的步骤把 TouchSwipe.js 包含到我们的插件文件中。

(1) 打开 TouchSwipe 的 GitHub 代码库：https://github.com/mattbryson/TouchSwipe-Jquery- Plugin。
(2) 下载整个代码库。
(3) 找到 jquery.touchSwipe.min.js 并复制其中的代码。
(4) 把代码粘贴到 plugins.js 文件中，位于 Bootstrap 插件之后。

(5) 保存。

插件就位了。接下来需要调用它。

C.3 调用TouchSwipe

需要写几行代码，命令 TouchSwipe 监听传送带上的轻扫事件，然后将其转换成 Bootstrap 的方法调用：`.carousel('prev')`和`.carousel('next')`。关于这些方法，大家可以参考 Bootstrap 的文档：http://getbootstrap.com/javascript/#carousel。

如果你感兴趣，也可以参考 TouchSwipe 的文档：http://labs.rampinteractive.co.uk/touchSwipe。

接下来的事很简单，只需下列几步。

(1) 打开项目中的 main.js 文件。

(2) 在其中添加如下代码：

```
//Enable swiping...
$(".carousel-inner").swipe( {
  //Generic swipe handler for all directions
  swipeRight:function(event, direction, distance, duration,
    fingerCount) {
    $(this).parent().carousel('prev');
  },
  swipeLeft: function() {
    $(this).parent().carousel('next');
  },
  //Default is 75px, set to 0 so any distance triggers swipe
  threshold:0
});
```

(3) 保存。

好了，如果你在触摸屏设备中测试网站，应该可以通过左、右轻扫来切换图片了。



就这么简单。代码不多，但很实用。

祝贺你，你的 Bootstrap 传送带支持手势了。

欢迎加入

图灵社区 iTuring.cn

——最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

优惠提示：现在购买电子书，读者将获赠书款20%的社区银子，可用于兑换纸质样书。

——最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

——最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、乐译、评选等多种活动，赢取积分和银子，积累个人声望。

Bootstrap实战

本书是目前市面上少见的实战类Bootstrap图书，全书通过5个真实、具体、鲜活，又有代表性的项目实例，讲解了Bootstrap的各种特性和用法。5个实例由浅入深，既各自独立，又环环相扣，丰富的代码，精美的插图，加上细致入微的解释，让读者极易上手，不知不觉中就能掌握所有重要概念，步入Bootstrap高手行列。本书讲解了以下5种类型的网站：个人作品网站、WordPress主题、企业门户页面、在线电子商务站点和单页营销网站。

除了令人惊艳的项目实例，本书还向读者介绍了很多实用插件、框架，以及前端开发的工作流程，这些知识和技术并不局限于Bootstrap。特别地，作者用相当篇幅介绍了LESS的基本原理和使用方法，让即使从未接触过LESS的人也能感觉游刃有余。此外，全书各章还分别介绍了HTML5 Boilerplate、Font Awesome、Respond.js、Masonry、Scroll-Spy，以及Roots的WordPress启动主题，让读者不必白手起家也能实现各种酷炫效果。



本书附录还介绍了为获得最快下载速度而对站点资源进行优化、实现响应式图片，以及为图片传送带添加手势的技术。作者提供的项目代码也非常完整、清晰，每一章都有相应的开始和完成文件夹，方便读者参考使用。

本书适合有一定HTML/CSS基础的开发人员和爱好者阅读学习，经验丰富的开发人员也可以把它当作参考。

[PACKT] PUBLISHING

图灵社区：iTuring.cn

热线：(010)51095186转600

分类建议 计算机/网页设计（Web前端）

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-38887-2

ISBN 978-7-115-38887-2

定价：49.00元